Universidade de Aveiro Departamento de Matemática
Ano 2009

André Alexandre Sebastião Marques

Métodos de Biclustering no Problema da Selecção de Genes

Universidade de Aveiro Departamento de Matemática
Ano 2009

André Alexandre Sebastião Marques

Métodos de Biclustering no Problema da Selecção de Genes

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática e Aplicações, com especialização em Ciências da Computação, realizada sob a orientação científica da Drª. Gladys Castillo Jordán, Professora auxiliar do Departamento de Matemática da Universidade de Aveiro, e sob a co-orientação científica da Drª. Adelaide de Fátima Baptista Valente Freitas, Professora auxiliar do Departamento de Matemática da Universidade de Aveiro.

Dedico este trabalho aos meus pais e avós pelo apoio incondicional de sempre.

o júri

presidente

Profª. Doutora Isabel Maria Simões Pereira
professora auxiliar da Universidade de Aveiro

Profª. Doutora Gladys Castillo Jordán
professora auxiliar da Universidade de Aveiro

Profª. Doutora Adelaide de Fátima Baptista Valente Freitas
professora auxiliar da Universidade de Aveiro

Prof. Doutor Miguel Francisco Rocha
professor auxiliar da Universidade do Minho

agradecimentos

Em primeiro lugar, gostaria de agradecer à Professora Adelaide e à Professora Gladys pela orientação, paciência, disponibilidade e motivação ao longo deste trabalho. Foi o vosso contributo que permitiu não só a conclusão como o enriquecimento deste trabalho.
Gostaria também de agradecer aos meus pais e avós pelo apoio incondicional de sempre e por contribuírem para a pessoa que sou hoje.
Um agradecimento muito especial também para a Maria Rosa e Veríssimo pela confiança que sempre depositaram em mim e por estarem sempre presentes ao longo da minha vida, tanto nos bons como nos maus momentos.
Gostaria também de deixar uma palavra de apreço para a minha madrinha e família.
Por último deixo aqui também um agradecimento especial à Lara pela motivação, apoio e paciência que sempre teve para comigo durante a realização deste trabalho.

palavras-chave

biclustering, aprendizagem automática, selecção de genes, mineração de dados, classificação supervisionada, classificação de cancro.

resumo

Com o desenvolvimento da tecnologia de *microarrays* nos últimos anos, tornou-se possível a monitorização simultânea do nível de expressão de milhares de genes, permitindo o avanço na investigação e identificação de genes associados a tecidos cancerígenos. Através do uso desta tecnologia, tornou-se um desafio extrair conhecimento relevante, especialmente do ponto de vista biológico, do enorme volume de dados acumulados a partir destas experiências. No entanto, devido à grande dimensionalidade, a análise deste tipo de dados torna-se inviável utilizando meios computacionais convencionais, pelo que técnicas de aprendizagem automática se apresentam como soluções bastante promissoras. Como provado em diferentes estudos, um pequeno subconjunto de genes altamente discriminativos é suficiente para construir classificadores bastante precisos. Consequentemente, o problema de selecção de genes é um dos problemas mais desafiantes no problema de classificação de cancro utilizando dados de microarrays. Com base em matrizes de níveis de expressão genética de genes sob diferentes condições experimentais, têm sido propostas metodologias de identificação de grupos homogéneos nestas usando métodos de *biclustering*. A aplicação de técnicas de *biclustering* pode ser uma mais valia para a determinação de genes que possam ser relevantes no diagnóstico de certos tipos de cancro.

No presente trabalho é apresentada uma abordagem ao problema de selecção de atributos baseada em métodos de *biclustering* combinados com uma heurística, que a partir dos *biclusters* resultantes, permite seleccionar um conjunto de atributos que melhor discriminam as classes. Estes métodos são posteriormente combinados com outras técnicas de selecção de atributos com o intuito de seleccionar subconjuntos de atributos (genes) altamente discriminativos presentes no conjunto de dados iniciais. Para a sua avaliação são apresentados os resultados de um estudo experimental sobre determinadas bases de dados pré-processadas a partir de uma base de dados brutos provenientes de um estudo sobre o cancro Lymphoma. Os resultados são analisados em termos da capacidade preditiva de um classificador de Máquinas de Suporte Vectorial, induzido por um subconjunto de genes seleccionado, através de validação cruzada *leave-one-out*.

keywords

biclustering, machine learning, genes selection, data mining, supervised classification, cancer classification.

abstract

During recent years, the development of microarray technology has made possible to monitor the expression levels of thousands of genes simultaneously. Particularly, these techniques allow the identification of genes associated with cancerous tissues. The enormous volume of data generated from microarray experiences allows us to extract relevant biological knowledge, thus contributing to improve cancer diagnosis. However, due to the large number of genes involved, the analysis of microarray data is not feasible using conventional data analysis techniques, so that, machine learning and data mining techniques have been successfully applied in these analysis. Moreover, as shown in different studies, a small subset of highly discriminative genes is sufficient to build highly accurate classifiers. Thus, gene selection is one of the most challenging problem in microarray data analysis. On the other hand, biclustering of the gene expressing data aims to identify homogeneous groups into the matrices of expression levels of genes under different experimental conditions. Therefore, biclustering methods can be applied to determine relevant subsets of genes in the diagnosis of certain types of cancer. In this dissertation we present an approach to the problem of feature subset selection (FSS) based on biclustering. We also propose a heuristic that uses the resulting biclusters as input and provide a subset with the most promising attributes as output. These methods are then combined with other FSS techniques in order to find a small subset of highly discriminative genes. Finally, we present an experimental study using different datasets resulting from pre-processing the row data generated from a study of the Lymphoma cancer. The results are analyzed in terms of the predictive capability of a support vector machine induced from the subset of relevant genes using leave-one-out cross-validation.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Contexto e motivação

Os *microarrays* de ADN[1] fazem parte de uma classe de biotecnologias onde é medido o nível de expressão de ADN ou de genes[2] de amostras de tecidos sob determinadas condições específicas, permitindo deste modo a monitorização dos níveis de expressão das células de milhares de genes em simultâneo. Um *microarray* de ADN [34] corresponde, geralmente, a uma lâmina de vidro ou silicone microscópica ou uma membrana de nylon no qual são afixadas sequências de milhares de genes, em posições específicas. O ADN é impresso, pontilhado ou sintetizado directamente no suporte dando um aspecto semelhante ao mostrado na Figura 1.1. Cada sequência de ADN, correspondente a um gene, é colocada num ponto específico do suporte. Os vários pontos formam uma matriz, cuja disposição ordenada e fixa permite, posteriormente, saber qual a sequência de ADN que se encontra em cada uma das posições.

Se a simples visualização dos dados provenientes das experiências com *microarrays* é, por si só, complexa, então a extracção de conhecimento biológico relevante representa um desafio ainda maior. Os dados de experiências de *microarrays*, são normalmente dispostos numa matriz de expressão genética, onde geralmente cada gene corresponde a uma linha e cada condição experimental (ex: classe de cada amostra) a uma coluna [6]. Assim, cada elemento da matriz contém o nível de expressão de um gene numa determinada condição - um valor real dado pelo logaritmo da quantidade relativa de ARNm[3] do gene numa determinada

[1] O ADN (ácido desoxirribonucleico) é o suporte universal da informação genética que define as características de cada organismo vivo. A unidade fundamental do ADN é o nucleótido, o qual resulta da ligação entre uma base azotada (A-adenina, G-Guanina, C-citosina, T-timina), uma pentose (desoxirribose) e um grupo fosfato.

[2] O gene é um segmento de um cromossoma a que corresponde um código distinto, uma informação para produzir uma determinada proteína ou controlar uma característica, por exemplo, a côr dos olhos.

[3] ARN é a sigla que designa o ácido ribonucleico (ou, em inglês, RNA, *ribonucleic acid*). O ARN é um polímero de nucleótidos, geralmente em cadeia simples, formado por moléculas de dimensões muito inferiores às do ADN. O ARN é constituído por uma pentose (Ribose), por um grupo fosfato e uma base

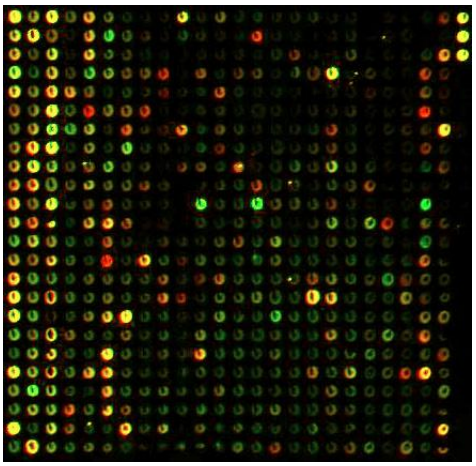

Figura 1.1: Exemplo de um *microarray* (retirada de [11]).

condição experimental.

A presente dissertação situa-se no contexto dos problemas de classificação supervisionada. Neste tipo de análise a partir de dados provenientes de experiências de *microarrays*, o objectivo é induzir um classificador (uma "função"), capaz de predizer com alta fiabilidade a classe de futuros exemplos. Para a construção deste classificador é necessário dispor de um conjunto de treino, constituído por exemplos previamente classificados. A classe toma valores num conjunto finito não ordenado.

Nos últimos anos, um dos problemas de classificação[4] de dados de expressão genética mais estudado é o problema de classificação de cancro. Isto fundamentalmente devido à existência de grandes volumes de dados gerados a partir de experiências de *microarrays* e disponibilizados on-line em grandes repositórios (ex: repositório de bases de dados de *microarrays* da Universidade de Standford [35]). Por outro lado, a classificação de cancro a partir de experiências de *microarrays* envolve conhecimento essencial para a resolução de problemas relacionados com o diagnóstico de diferentes tipos de tumores assim como a descoberta de novos medicamentos para o seu tratamento [15]. No problema da classificação de cancro os dados de *microarrays* são organizados em matrizes de níveis de expressão genética de dimensão $n \times m$, onde n representa o número de genes e m o número de condições experimentais (amostras de tecidos com a informação da classe, onde a classe, pode ser, por exemplo se um tecido é ou não canceroso).

Muita da informação presente nas bases de dados obtidas de experiências de *microarrays* de ADN é redundante, irrelevante e contém bastante "ruído" o que pode afectar a qualidade preditiva de um classificador quando induzido por esta. Além disso, os dados de

azotada (nitrogenada) que pode ser adenina (A), guanina (G), citosina (C) e Uracila (U). O ARNm (ARN Mensageiro) é um tipo de ARN com a função de transportar as informações do código genético do ADN para o citoplasma, ou seja, determina as sequências dos aminoácidos na construção das proteínas.

[4] A partir deste momento referir-se-á a classificação supervisionada apenas por classificação.

microarrays contêm informação sobre um número elevado de genes (geralmente na ordem dos milhares) e em contraste o número de amostras é, como regra, reduzido (não ultrapassando a centena). Como tem sido provado em diferentes estudos [15, 16], um pequeno subconjunto de genes altamente discriminativos é suficiente para construir classificadores bastante precisos. Devido a este facto, a selecção de genes relevantes é um dos problemas mais críticos e que mais atenção tem merecido por parte de vários investigadores em estudos baseados em dados de microarrays [12, 13, 14, 15]. Num contexto mais geral de problemas de classificação supervisionada, a tarefa anteriormente descrita é denominada selecção de conjunto de atributos (traduzido do inglês de *feature subset selection*).

Dos pontos de vista biológico e clínico, encontrar um número reduzido de genes importantes pode ajudar os investigadores a concentrar-se nesses genes e investigar os mecanismos de desenvolvimento e tratamento de cancro. Pode ainda reduzir os custos de testes laboratoriais, uma vez que o paciente necessita apenas de ser testado para certos e determinados genes, em vez de milhares deles. Além disso, torna-se possível obter métodos simples para os médicos fazerem diagnósticos mesmo sem usarem um classificador ou um computador, por exemplo, em casos em que apenas um ou dois genes sejam necessários para a classificação [12].

Uma vez que os dados de expressão genética estão normalmente organizados numa matriz (genes por condições experimentais ou vice-versa), têm sido recentemente propostas metodologias para a identificação de padrões, de grupos de dados homogéneos presentes nas matrizes. Uma dessas metodologias é o *biclustering*.

O *biclustering* permite obter um subconjunto do conjunto de dados iniciais através de técnicas que agrupam, simultaneamente, os genes e as condições da matriz de expressão genética, podendo ser uma mais valia para a determinação de genes que possam ser relevantes para o diagnóstico de certos tipos de cancro [31]. Quando esta técnica é realizada com alta fiabilidade, é possível não só diagnosticar as condições representadas por grupos de amostra, mas também identificar os genes responsável por estas.

Existem alguns trabalhos que propõem usar diferentes técnicas de *biclustering* com o intuito de identificar os genes mais relevantes usando dados de *microarrays* [36, 37, 38]. No presente trabalho é estudado e avaliado o algoritmo de *biclustering* ISA (Iterative Signature Algorithm) proposto por Bergmann et al [6] e uma variante deste, desenvolvida por Freitas et al [10], no contexto do problema de selecção de atributos em dados de expressão genética.

1.2 Objectivos e contribuições

O principal objectivo da presente dissertação é avaliar diferentes técnicas de selecção de genes baseadas na extracção de *biclusters* em matrizes de dados de expressão genética utilizando o algoritmo de *biclustering* ISA. As metodologias propostas são avaliadas através do desempenho preditivo de classificadores induzidos de dados de microarrays.

As contribuições fundamentais desta tese são:

1. Apresentar uma abordagem ao problema de selecção de atributos baseada em métodos de *biclustering* utilizando várias configurações de parâmetros de entrada do ISA.

2. Apresentar uma heurística que, a partir do conjunto de *biclusters* resultantes da aplicação do ISA, permite seleccionar um conjunto de atributos que melhor discriminam as classes. Esta redução do conjunto inicial de atributos permite reduzir significativamente os custos computacionais de um algoritmo *wrapper* de selecção de atributos, implementado a seguir, para determinar finalmente um subconjunto muito reduzido de genes altamente discriminativos.

3. Apresentar um estudo experimental para avaliação do desempenho preditivo de máquinas de suporte vectorial quando induzidas por subconjuntos de atributos obtidos através da aplicação dos métodos propostos ao conjunto de dados iniciais.

1.3 Organização

Esta dissertação é constituída por 6 capítulos, sendo este primeiro uma introdução ao estudo que posteriormente irá ser realizado.

Em seguida, no Capítulo 2, faz-se uma abordagem ao problema de selecção de atributos de modo geral, assim como uma breve descrição e explicação dos métodos de selecção mais conhecidos e que normalmente são utilizados no contexto do problema de classificação de cancro.

No Capítulo 3 é explicado um dos focos principais desta tese, ou seja, a técnica de *biclustering*. Será feita uma descrição pormenorizada de diferentes tipos de *biclusters* e uma descrição sumária dos diferentes tipos de algoritmos de *biclustering* que podem ser encontrados.

Depois, no Capítulo 4, será apresentado o principal algoritmo de *biclustering* estudado neste trabalho, denominado por ISA.

O Capítulo 5 é dedicado à descrição de todo o estudo experimental realizado para avaliar os métodos propostos nesta tese, assim como a apresentação dos resultados obtidos e uma discussão sobre os mesmos.

Por fim, no Capítulo 6, serão descritas as conclusões acerca do trabalho realizado e dos resultados obtidos, em conjunto com algumas sugestões que podem ser relevantes para um trabalho futuro.

Capítulo 2

O problema da selecção de genes

2.1 Introdução

O perfil das expressões de genes tem sido bastante utilizado no estudo de assinaturas moleculares de diversas doenças e no desenvolvimento de diagnósticos moleculares. Alguns genes, quando associados a outros, podem ser relevantes para a classificação de uma nova amostra e, no entanto, individualmente inúteis para o problema em estudo. Devido a este facto, a selecção de genes relevantes para a classificação de uma nova amostra é uma tarefa comum na maioria dos estudos de expressão genética, onde os investigadores tentam identificar o menor conjunto possível de genes. O objectivo é permitir atingir um bom desempenho de preditivo em futuros exemplos (por exemplo, para futura utilização com fins de diagnóstico na prática clínica).

Apesar do código genético humano estar "quase" terminado, a sua análise ainda agora começou. Além de sequências de informação, novos *microarrays* de DNA estão constantemente a surgir, oferecendo grandes quantidades de informações sobre a vida interior de uma célula. O novo desafio é agora avaliar esses gigantes fluxos de dados com o intuito de extrair padrões que possam ser relevantes e úteis para os diagnósticos de determinadas tipos de doenças e descoberta de novos medicamentos.

Segundo [15] existem quatro principais desafios. O primeiro resulta no facto do espaço de atributos ser enorme (geralmente na ordem das centenas de milhares de genes por amostra), provocando o problema das amostras serem vistas como pontos muito escassos num espaço de grandes dimensões quando feito o seu mapeamento para pontos no espaço de atributos. A maioria dos algoritmos de classificação existentes não estão desenvolvidos para este tipo de características, o que provoca o acontecimento de problemas de *overfitting*[1] assim como problemas relacionados com o tempo de computação.

O segundo desafio resulta da presença de "ruído" no conjunto de dados. Este "ruído"

[1] formação de um classificador sobreajustado a um conjunto de dados complexos que apresenta um mau desempenho preditivo.

pode ser categorizado em biológico e técnico [15]. Assim, o "ruído" biológico é introduzido pelos genes que não apresentem relevância na tarefa de classificação. Como o número de amostras é muito reduzido em comparação com o número elevado de genes, a presença de "ruído" biológico pode afectar negativamente o desempenho preditivo do classificador. Por outro lado, as experiências de *microarrays* envolvem muitos passos: impressão do ADN-complementar, extracção de ARN mensageiro, marcação, hibridação, *scanning*, processamento de imagem, entre outras. Cada uma dessas etapas pode introduzir "ruído" (variabilidade nas intensidades medidas) e afectar a qualidade dos dados. Neste caso, é preciso aplicar métodos de limpeza (correcção de *background*) e transformação (normalização) com o intuito de refinar os dados brutos para eliminar o "ruido" técnico mas mantendo as variações biológicas intrínsecas.

O terceiro desafio é motivado pela existência de um grande número de genes que são realmente irrelevantes para a tarefa de classificação. Como foi provado em diversos estudos experimentais [13, 14, 15, 16, 24], apenas pouquíssimos genes (podem ser até 2) são suficientes para poder discriminar correctamente entre diferentes tipos de tumores. A melhor forma de lidar com este problema é implementar um procedimento de selecção de genes, explicado em detalhe neste capítulo.

O último desafio está directamente relacionado com o problema em estudo sobre classificação de cancro. Neste caso, não só a precisão da classificação é importante como também a interpretação em termos biológicos de toda a informação que possa ser descoberta. Por exemplo, durante o processo de classificação pode ser adquirida informação sobre genes que actuando em grupos podem relevar a existência de tecidos cancerosos. Toda esta informação ajuda os especialistas a perceber melhor o genoma humano e como diferentes genes interagem entre si.

Muitos genes são fortemente regulados e apenas transcritos em determinados momentos, determinadas condições e em certos tipos de célula. As experiências de *microarrays* permitem medir simultaneamente o nível de expressão do mRNA de milhares de genes numa célula. Através da comparação dos perfis de expressão dos vários tipos de tecidos é possivel, na maioria das vezes, descobrir genes que melhor expliquem determinadas perturbações, ou anomalias, ou até mesmo ajudar a descobrir como certo tipo de cancro se está a desenvolver.

Dada uma série de conjuntos experimentais de *microarrays* de um tecido específico sob diferentes condições, o objectivo será então descobrir os genes que estejam expressos de forma diferente sob essas condições. Por outras palavras, pretende-se descobrir genes que melhor expliquem os efeitos dessas condições. A tarefa de selecção de genes é um caso particular do problema de selecção de conjuntos de atributos (do termo em inglês *feature subset selection*), uma das etapas de pré-processamento mais críticas e importantes em problemas de classificação supervisionada. A selecção de atributos pretende identificar quais os atributos que melhor contribuem para a discriminação das classes, ou seja, pretende-se seleccionar os atributos que tenham um maior impacto no desempenho preditivo de classificadores induzidos de dados, e deixar de fora aqueles que não tenham impacto, ou cujo impacto seja insignificante. Uma vez identificados, estes atributos podem ser usados para construir um

classificador com um melhor desempenho preditivo e classificar futuros exemplos para os quais a sua classificação é desconhecida.

Em seguida serão descritos alguns dos principais métodos de selecção de atributos.

2.2 Selecção de atributos em classificação supervisionada

A selecção de atributos é um dos problemas mais explorados nas áreas de aprendizagem automática, mineração de dados, reconhecimento de padrões, entre outros. Neste tipo de problemas vai-se reduzir o número de atributos, que são usados para caracterizar um conjunto de dados, por forma a melhorar o desempenho de um algoritmo de aprendizagem durante uma tarefa particular. Reduzir o número de atributos significa reduzir a dimensionalidade do espaço dos atributos. A aprendizagem supervisionada tem como objectivo induzir um classificador que possibilite predizer a classe de um novo exemplo apenas caracterizado pelos valores dos seus atributos, isto é, pretende-se classificar um novo exemplo com base num conjunto exemplos sobre os quais se tem conhecimento das respectivas classes e dos correspondentes valores dos atributos.

Uma selecção de atributos bem sucedida conduz à redução da complexidade de um problema, facilitando a visualização e interpretação dos dados. Como já se referiu anteriormente, atributos irrelevantes ou redundantes, em função de características específicas do algoritmo de aprendizagem, podem influenciar negativamente o desempenho preditivo deste. Aplicando este tipo de métodos normalmente consegue-se eliminar grande parte deste tipo de atributos, melhorando não só o esforço computacional como, consequentemente, o tempo de execução de um modelo de classificação visto que este passa a ser induzido de menos atributos, mas também o seu desempenho preditivo. No entanto este método pode introduzir um aumento considerável do custo computacional na tarefa de aprendizagem supervisionada, uma vez que, é adicionada uma camada adicional de complexidade no processo de encontrar um modelo de classificação.

Figura 2.1: Esquema geral do funcionamento de um método de selecção de atributos.

O problema de selecção de atributos pode ser visto como um problema de busca onde cada estado no espaço de busca representa um possível subconjunto de atributos (Figura 2.1). Está provado que este problema é um problema de optimização discreta NP-difícil

[39], ou seja, normalmente para a sua resolução recorre-se à implementação de algoritmos heurísticos que permitam determinar uma solução aproximada da solução óptima.

Assim, para procurar o melhor subconjunto de atributos é necessário definir:

1. uma solução inicial (um subconjunto inicial de atributos)
2. uma função objectivo, para guiar a busca no espaço de possíveis subconjuntos de atributos, em conjunto com uma medida de avaliação
3. uma estratégia de busca (ex: *brute-force*, *hill-climbing*, *best-first search*, ...)
4. um critério de paragem

Como regra, normalmente os critérios de paragem mais usados são: *(i)* parar se é atingido um número máximo de iterações; *ii)* parar se o subconjunto de atributos contém um número máximo de elementos; *iii)* parar se o valor da função objectivo (medida usada) ultrapassa o valor de um *threshold*[2] dado.

No contexto da classificação supervisionada, as técnicas de selecção de atributos podem ser organizadas em duas categorias:

1. métodos baseados em filtros
2. métodos *wrapper*
3. métodos incorporados (*embedded*)

2.2.1 Métodos baseados em filtro

Um método baseado em filtro selecciona os atributos relevantes considerando apenas as propriedades intrínsecas dos dados. Como regra, é definido um *threshold* e a cada atributo é atribuído um *score*[3] que representa o seu grau de relevância. Em seguida todos os atributos com valor do *score* superior ao do *threshold* são considerados como relevantes sendo os restantes removidos. Finalmente, o conjunto de atributos seleccionados é utilizado para compor o conjunto de treino que vai ser usado como parâmetro de entrada de um algoritmo de aprendizagem.

Este tipo de técnica tem a vantagem de poder ser facilmente implementada em conjuntos de dados de grande dimensão uma vez que é computacionalmente simples e rápida, e funciona independentemente do algoritmo de classificação. Apenas é necessário ser efectuada uma vez, podendo-se depois avaliar diferentes classificadores.

Uma desvantagem dos métodos de filtro é o facto de ignorarem a iteracção com o processo de indução do classificador e de, na sua maioria, as técnicas propostas serem univariadas,

[2] termo inglês, significa um valor, parâmetro limiar

[3] termo inglês, neste contexto é representativo do peso, da importância de um atributo

isto é, ignoram as dependências entre os atributos. Este factor pode levar à obtenção de um pior desempenho preditivo dos classificadores induzidos quando comparados com outras técnicas de selecção de atributos.

De forma a contornar este problema, um número de técnicas de filtro multivariadas, que têm em conta a possivel correlação entre atributos e dos atributos com a classe, têm sido desenvolvidas. Uma das técnicas de filtros mais utilizada é baseada numa medida denominada CFS (*Correlation-based Feature Selection* [5]). Esta medida avalia a importância de um subconjunto de atributos considerando a "utilidade" de cada atributo individual para predizer o valor da classe assim como o grau de correlação entre os atributos. A ideia em que se basea é a seguinte: "bons subconjuntos de atributos contêm atributos muito correlacionados com a classe, mas pouco correlacionados entre si".

2.2.2 Métodos *wrapper*

Os métodos *wrapper* utilizam o algoritmo de aprendizagem, depois utilizado para induzir um classificador, de forma a avaliar o desempenho de cada subconjunto de atributos candidato. Durante o processo de busca (ver Figura 2.2), cada subconjunto de atributos candidato é avaliado, usando como medida o resultado da estimação do desempenho de um classificador induzido por um conjunto de treino que contêm para cada exemplo apenas os valores dos atributos deste subconjunto e a classe correspondente.

Assim, a avaliação de um determinado subconjunto de atributos é feita testando o modelo de classificação induzido usando apenas os atributos desse subconjunto.

A vantagem da utilização dos métodos *wrapper* está na interacção entre o processo de selecção de atributos e o processo de selecção do modelo de classificação implementado pelo algoritmo de aprendizagem [14]. Como desvantagens, importa realçar o facto de existir uma maior probabilidade do que nos métodos de filtro de *overfitting* e de serem computacionalmente mais intensivos, especialmente se a construção do modelo de classificação tiver um elevado custo computacional.

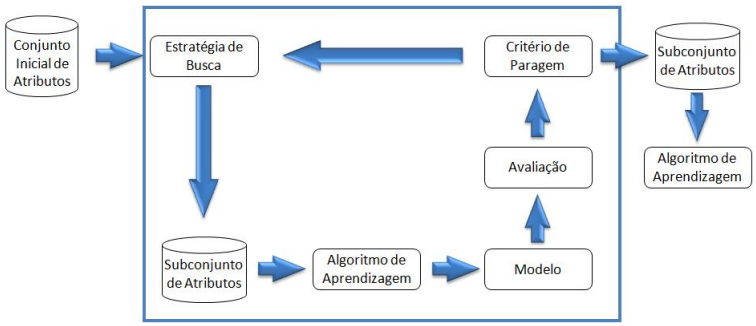

Figura 2.2: Esquema geral de selecção de atributos dos métodos *wrapper*.

2.2.3 Métodos incorporados

Nos metódos incorporados (*embedded methods*), a busca por um subconjunto óptimo de atributos está "embutida" no processo de construção de um classificador, ou seja, o próprio algoritmo de aprendizagem decide quais os atributos a utilizar. Assim como os métodos *wrapper*, estes métodos têm como grande vantagem a interacção com os algoritmos de aprendizagem na indução do classificador, mas exigem muito menos esforço computacional. Um dos exemplos mais evidentes são os algoritmos de indução de árvores de decisão pois durante o processo de construção do modelo vão seleccionando os atributos mais relevantes e que melhor discriminam as classes para nós da árvore.

2.3 Selecção de genes em dados de *microarrays*

A análise de dados de expressão genética obtidos a partir de experiências de *microarrays* de ADN constitui um grande desafio em termos de técnicas computacionais, devido à sua grande dimensionalidade (ao elevado número de genes) e ao tamanho reduzido das suas amostras. A fim de lidar com estas características particulares, as técnicas de selecção de atributos tem sido aplicadas com sucesso [40, 41] neste contexto.

Devido a essa grande dimensão do espaço de atributos existente na maioria das análises com dados provenientes de experiências de *microarrays*, e consequentemente ao esforço computacional exigido, as técnicas de filtro univariadas de selecção de atributos [14], técnicas que não têm em consideração possíveis dependências entre atributos, são as mais utilizadas.

Segundo [14], a preferência por este tipo de técnicas pode ser explicada pelos seguintes motivos:

- o *output* determinado por técnicas de filtro univariadas é intuitivo e de fácil compreensão;
- o *ranking* de genes obtido pode cumprir os objectivos e expectativas que os biotécnicos pretendem para em seguida validarem os resultados em experiências laboratoriais.
- não exigem o esforço computacional extra que é necessário em técnicas de filtro multivariadas.

Neste tipo de técnicas, os atributos são ordenados segundo a sua correlação com a respectiva classe, não entrando em consideração a sua possível correlação com outros atributos. Os genes seleccionados apresentam uma grande correlação individual com a classe, mas juntos podem não apresentar a melhor performance de classificação, enquanto que genes que se complementam entre si para determinar a classe podem não ser seleccionados se não apresentarem um grande coeficiente individual de correlação.

Os métodos de selecção de atributos univariados têm certas restrições e podem levar à obtenção de classificadores menos exactos, por exemplo, devido ao facto de não terem em

consideração a interacção/dependência entre genes. Sendo assim, alguns investigadores propuseram técnicas que permitem utilizar essa correlação entre genes.

A ideia principal consiste em descobrir um subconjunto de atributos que permitam determinar o melhor classificador possivel, não tendo em conta apenas a correlação individual dos genes com a classe, mas sim a correlação entre genes e entre um subconjunto de genes e a classe.

A aplicação de métodos de filtro multivariados vão desde simples interacções bivariadas, a soluções mais avançadas, que exploram maiores níveis de correlação, tais como o CFS.

A selecção de atributos utilizando métodos *wrapper* ou incorporados oferecem uma forma alternativa de implementar uma selecção de genes multivariada, incorporando o classificador na procura e permitindo desta forma um melhor desempenho preditivo com os genes seleccionados. No contexto da análise com dados de *microarrays*, a maioria dos métodos *wrapper* utiliza heurísticas de procura aleatória com base na população de genes inicial. Outra característica dos métodos *wrapper* é a função utilizada para avaliar cada subconjunto de genes encontrado, sendo a taxa de erro 0-1 a mais utilizada. Esta taxa mede o custo de atribuir a um exemplo uma classe predicta pelo algoritmo de classificação, concretamente se a classificação do exemplo for correctamente feita pelo algoritmo, então não existe custo, caso contrário existe um custo/perda de 1.

Em grande parte devido ao enorme esforço computacional dos métodos *wrapper* e à menor importância dada aos métodos incorporados até agora, estas técnicas, não recebem tanto interesse como os métodos de filtro. No entanto, uma boa práctica a seguir nestes casos é reduzir inicialmente o número de atributos utilizando uma técnica de filtro univariada e em seguida aplicar um método *wrapper* ou incorporado, para melhores resultados e um não muito elevado esforço computacional.

Avaliando o peso que um atributo individual pode contribuir para a distinção de um atributo num conjunto de dados é possível produzir um simples *ranking* para classificação desse atributo. Por conseguinte, avaliando o peso de cada gene de um determinado conjunto, é possível estabelecer um critério de *ranking*.

Em [16] são referidos e explicados processos que podem ser utilizados na selecção de atributos, utilizando um critério de *ranking*, através da utilização de pesos provenientes de um classificador MSV linear.

Um procedimento iterativo de selecção de atributos com base neste critério terá a seguinte estrutura:

1. Testar o classificador;

2. Determinar o valor da característica para cada um dos atributos. Ordenar os atributos de acordo com esse valor;

3. Remover o(s) atributo(s) com menor valor segundo o critério de ordenação.

Este procedimento em [16] é denominado por *eliminação recursiva de atributos usando máquinas de suporte vectorial*[4] (ERA-MSV).

[4]Traduzido do inglês "Support vector machines recursive feature eliminação (SVM-RFE)"

Capítulo 3

Biclustering

3.1 Introdução

A expressão de genes tem sido estabelecida na última década como uma técnica standard para obtenção de impressões digitais de tecidos ou células sob diferentes condições biológicas. Baseada na disponibilidade de sequências de genomas, técnicas como, por exemplo, a tecnologia dos *microarrays* têm sido impulsionadas de forma a puderem contribuir de maneira importante para a pesquisa genética funcional de diferentes organismos, desde bactérias até ao homem, em situações normais e patológicas (cancro, doenças auto-imunes, doenças degenerativas entre outras). É através deste tipo de técnicas que é possível a obtenção de dados numéricos, sendo desafiador e extremamente difícil extrair conhecimentos biológicos úteis e relevantes. A análise destes dados pode servir para comparar a expressão genética entre dois tipos celulares distintos, por exemplo, entre tecidos saudáveis e tecidos doentes ou para examinar as alterações na expressão genética, por exemplo, durante o desenvolvimento embrionário ou, ainda que de forma indirecta, para classificação de novos genes com prévio conhecimento de classificação de outros genes, entre muitos outros.

Normalmente a expressão de genes é disposta através de uma base de dados matricial, onde cada gene corresponde a uma linha e cada coluna a uma determinada condição. Cada elemento desta matriz é um número real representativo do nível de expressão de um determinado gene segundo uma condição específica. Dado um conjunto de níveis de expressão de genes assim organizado, um objectivo de análise é identificar padrões de subconjuntos de genes e condições biologicamente relevantes. Este objectivo pode ser divido em duas partes: análise de padrões de condições experimentais por comparação de colunas da matriz, ou análise de padrões de genes por comparação entre linhas. As duas análises anteriormente referidas são conhecidas como *clustering*, Figura 3.1.

Contudo, a aplicação de algoritmos de *clustering* em bases de dados de expressão de genes apresentam certas dificuldades. Os algoritmos de *clustering* separam os elementos em grupos disjuntos e, portanto, um gene ou condição não poderá pertencer a mais do que um *cluster*. Muitos dos padrões identificados aquando da utilização desta técnica, deveriam ser

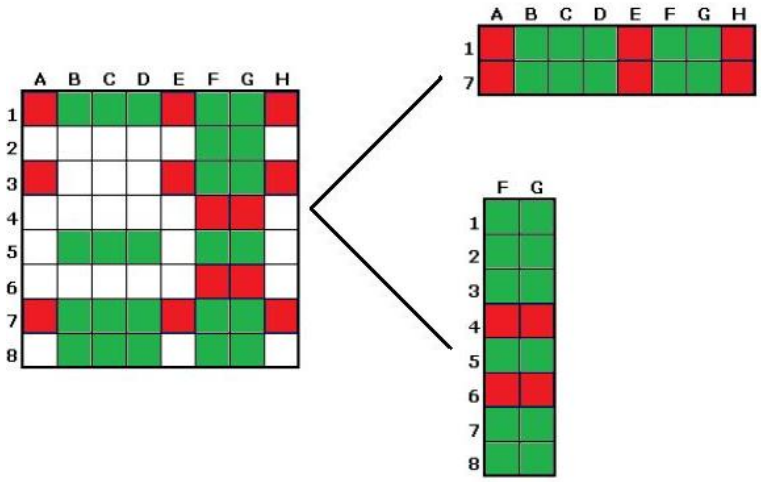

Figura 3.1: Exemplo de dois *clusters* existentes na matriz dados (à esquerda): um obtido por *clustering* de linhas e outro por *clustering* de colunas.

comuns a um determinado número de genes e a várias condições. Outro factor negativo no *clustering* é agrupar os genes de acordo com o seu comportamento perante todas as condições. Recentes investigações biológicas relativas a processos celulares evidenciam que os genes pode ser co-expressos para um conjunto de condições experimentais, apresentando um comportamento completamente independente para um outro conjunto de condições. Deste modo conclui-se que a técnica de *clustering* pode não ser a mais fiável em diversas situações [2]. É aqui que entram o métodos principais que dão titulo a este capitulo, ou seja, os métodos de *biclustering*.

O termo *biclustering* foi introduzido pela primeira vez por Cheng e Church [1] na análise de dados de expressão genética. Os métodos de *clustering* são aplicados ou a linhas ou a colunas de uma certa matriz de dados, mas nunca às duas dimensões da matriz simultaneamente. Ou seja, enquanto métodos de *clustering* resultarão num modelo global, o *biclustering* permitirá obter um modelo local dos dados através de estratégias que agrupam, simultaneamente, os elementos de uma matriz tanto nas linhas como nas colunas, Figura 3.2.

Ao utilizar algoritmos de *clustering* cada gene agrupado num *cluster* é definido utilizando todas as condições, e portanto cada condição contida num *cluster* é caracterizada pela actividade de todos os genes. Já num *bicluster* cada gene contido neste foi seleccionado utilizando apenas um subconjunto de condições e cada condição seleccionada utilizando um subconjunto de genes. Os algoritmos de *biclustering* permitem seleccionar, identificar grupos de genes com padrões similares sobre um subconjunto de condições experimentais. A técnica de *biclustering* tem-se mostrado de uma importância extrema em diversas situações, tais como, quando apenas um pequeno conjunto de genes participa num processo celular de interesse ou, um processo celular está activo apenas num pequeno conjunto de condições.

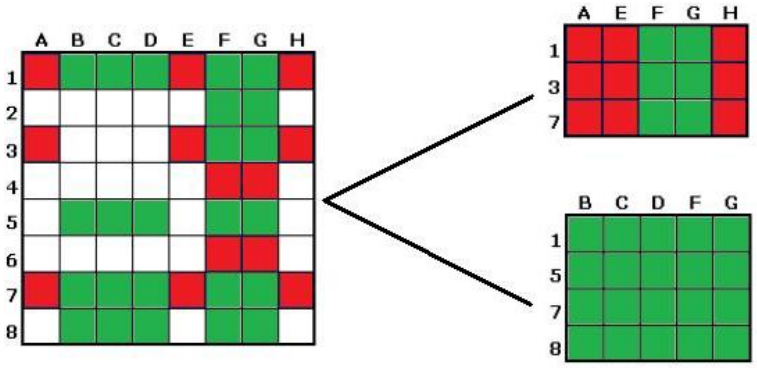

Figura 3.2: Exemplo de dois *biclusters* existentes na matriz dados considerada na Figura anterior.

Em suma: um *cluster* de genes deve ser definido de acordo com apenas um restrito subconjunto de condições; um *cluster* de condições deve ser definido de acordo com apenas um restrito subconjunto de genes; um gene ou condição não tem que estar obrigatoriamente num *bicluster* ou apenas num *bicluster*, pode aliás estar em vários.

3.2 Definições e formulação do problema

Considere-se, como caso geral, uma matriz de dados X, $n \times m$, definida por um conjunto de linhas $L = \{l_1, l_2, \ldots, l_n\}$ e um conjunto de colunas $C = \{c_1, c_2, \ldots, c_m\}$. Cada elemento x_{ij} corresponde a um valor que representa a relação entre a linha l_i e a coluna c_j (Tabela 3.1). Considerando que $I \subseteq L$ e $J \subseteq C$ são subconjuntos de linhas e colunas, X_{IJ} denota a submatriz de X que contém apenas os elementos x_{ij} que pertencem à submatriz com o conjunto de linhas I e o conjunto de colunas J. Dada a matriz X, um *cluster* de linhas

	c_1	$\ldots$	c_j	$\ldots$	c_m
l_1	x_{11}	$\ldots$	x_{1j}	$\ldots$	x_{1m}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
l_i	x_{i1}	$\ldots$	x_{ij}	$\ldots$	x_{im}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
l_n	x_{n1}	$\ldots$	x_{nj}	$\ldots$	x_{nm}

Tabela 3.1: Matriz X de Dados de Expressão de Genes.

é definido por um subconjunto de linhas com um comportamento semelhante em todo o conjunto de colunas. Ou seja, um *cluster* de linhas X_{IC} é dado por um subconjunto I de

k linhas definido sobre todo o conjunto de colunas C em que $k \leq n$. Em suma, um *cluster* de linhas X_{IC} pode assim ser definido como uma submatriz de X, $k \times m$.

De forma análoga, um *cluster* de colunas X_{LJ} pode ser definido por uma submatriz de X, $n \times s$ ($s \leq m$). Ou seja, um *cluster* de colunas X_{LJ} é dado por um subconjunto J de s colunas definido sobre todo o conjunto de linhas L em que $J \subseteq C$ e $s \leq m$. O subconjunto s de colunas tem um comportamento semelhante em todo o conjunto de linhas.

Um *bicluster* é definido por um subconjunto de linhas que exibem um comportamento semelhante em todo um subconjunto de colunas, e vice-versa. O *bicluster* X_{IJ} é dado por um subconjunto de linhas $I = \{i_1, \ldots, i_k\}$ ($I \subseteq L$ e $k \leq n$) e um subconjunto de colunas $J = \{j_1, \ldots, j_s\}$ ($J \subseteq C$ e $s \leq m$). Ou seja, um *bicluster* X_{IJ} pode ser definido como uma submatriz de X com k linhas e s colunas.

3.2.1 Grafos bipartidos com pesos e matrizes de dados

Madeira e Oliveira [2] fazem referência a uma relação interessante entre teoria de grafos e matrizes de dados. Uma matriz de dados pode ser vista como um grafo bipartido com pesos. Um grafo $G = (V, E)$, onde V é o conjunto de vértices e E o conjunto de arestas, diz-se bipartido se os seus vértices podem ser divididos em dois conjuntos, C_1 e C_2, tais que toda a aresta em E tem exactamente um vértice em C_1 e outro em C_2, e $V = C_1 \cup C_2$. Desta forma, a matriz de dados X pode ser vista como um grafo bipartido com pesos onde cada nodo $l_i \in L$ corresponde a uma linha e cada nodo $c_j \in C$ corresponde a uma coluna. A aresta entre o nodo l_i e o nodo c_j tem um peso x_{ij}, denotando o elemento da matriz na posição de intersecção entre a linha i e a coluna j.

3.2.2 Complexidade

Apesar da complexidade do problema de *biclustering* poder depender da sua formulação exacta, mais especificamente, da função objectivo usada para identificar um determinado *bicluster*, este é um problema que, em quase todas as suas variantes, é NP-Completo[3]. Numa versão simplificada, tomando a matriz de dados X como uma matriz binária, um *bicluster* corresponde a um biclique[1] no grafo bipartido que lhe corresponde (ver Figura 3.3). Encontrar um *bicluster* de tamanho máximo é equivalente a procurar uma biclique máxima, com o maior número de arestas num grafo bipartido, o que constitui um problema NP-Completo [3]. Em casos mais complexos, em que a matriz de dados X assume valores numéricos reais, na avaliação de qualidade de um determinado *bicluster*, a complexidade do problema nunca é inferior ao caso da matriz binária.

Devido a esta complexidade, a maioria dos algoritmos propostos para o problema de *biclustering* recorre a abordagens heurísticas. Na grande parte dos casos, essas heurísticas são

[1] grafo bipartido completo em que cada vértice de um conjunto X está ligado a todos os vértices de um outro conjunto Y

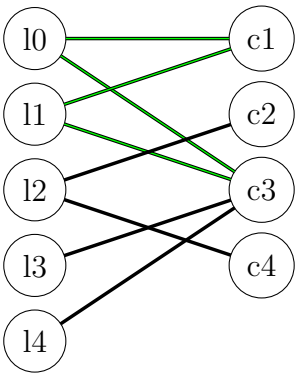

Figura 3.3: Exemplo(a verde) de um Bicluster num Grafo Bipartido.

precedidas por um passo de normalização aplicado à matriz de dados de forma a evidenciar padrões de interesse.

3.3 Algoritmos de *biclustering*

Uma forma simples (mas pouco eficaz) de obter um *bicluster*, a partir de um conjunto de dados representado na forma matricial, é utilizar métodos tradicionais de *clustering* ao longo das linhas e colunas separadamente e combinar depois os resultados obtidos de forma a obter uma submatriz da matriz inicial que satisfaça um conjunto de parâmetros pré-definidos. Actualmente já existem um número significativo de algoritmos para este tipo de problemas. Uma das primeiras formulações foi introduzida em 1972 por Hartigan [4] com o objectivo de obter *biclusters* com valores constantes reordenando as linhas e colunas da matriz de dados de forma a produzir uma matriz com blocos homógeneos. Nos últimos anos têm sido propostos vários algoritmos com a finalidade de identificar agrupamentos cada vez mais complexos. Na área da biologia computacional, tais algoritmos têm sido propostos principalmente para o tratamento de dados de expressão de genes obtidos de experiências de *microarrays*. Cheng e Church [1] foram os primeiros a propor um algoritmo para este tipo de problemas.

Madeira e Oliveira [2] classificam os algoritmos de *biclustering* segundo as seguintes quatro dimensões de análise:

- O tipo de *biclusters* que conseguem encontrar (Secção 3.3.1);
- A estrutura do(s) *bicluster(s)* produzido(s) (Secção 3.3.2);
- O algoritmo específico usado para identificar cada *bicluster* (Secção 3.3.3);
- O domínio da aplicação de cada algoritmo.

Nas próximas subsecções serão descritas, mais detalhadamente, algumas dessas dimensões.

3.3.1 Tipos de *biclusters*

Madeira e Oliveira [2] identificam quatro tipos de *biclusters* possíveis de encontrar numa matriz de dados (ver Tabela 3.2). São estes:

1. *Biclusters* com valores constantes.
2. *Biclusters* com valores constantes em linhas ou colunas.
3. *Biclusters* com valores coerentes.
4. *Biclusters* com evoluções coerentes.

Os algoritmos de *biclustering* mais simples identificam submatrizes de valores constantes, Tabela 3.2(a). Neste caso, um *bicluster* perfeito será uma submatriz em que todos os elementos são iguais a um valor constante. No entanto, em dados reais, estes *biclusters* raramente se encontram devido ao ruído normalmente existente nos dados. Outros algoritmos, que mais facilmente identificam conjuntos de *biclusters* em dados reais, tentam identificar submatrizes cujos valores sejam constantes nas linhas, Tabela 3.2(b), ou submatrizes cujos valores sejam constantes nas colunas, Tabela 3.2(c).

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

(a)

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

(b)

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

(c)

1	2	5	0
2	3	6	1
4	5	8	3
5	6	9	4

(d)

1	4	2	6
2	8	4	12
3	12	6	18
4	16	8	24

(e)

S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1
S1	S1	S1	S1

(f)

S1	S1	S1	S1
S2	S2	S2	S2
S3	S3	S3	S3
S4	S4	S4	S4

(g)

S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4
S1	S2	S3	S4

(h)

↗	↗	↘	↘
↘	↘	↗	↗
↗	↗	↘	↘
↘	↘	↗	↗

(i)

Tabela 3.2: Tipos de *bicluster*: (a) *bicluster* constante; (b) com linhas constantes; (c) com colunas constantes; (d) com valores coerentes modelo aditivo; (e) com valores coerentes modelo multiplicativo; (f) evolução global coerente; (g) evoluções coerentes nas linhas; (h) evoluções coerentes nas colunas; (i) evoluções coerentes em linhas e colunas

.

Repare-se que um *bicluster* com linhas constantes identifica um subconjunto de genes num determinado subconjunto de condições, permitindo que os níveis de expressão sejam diferentes de gene para gene. De forma análoga, um *bicluster* com colunas constantes identifica um subconjunto de condições no qual um conjunto de genes apresenta valores de expressão semelhantes assumindo que os seus valores de expressão podem variar de condição para condição. Caso interesse encontrar *biclusters* mais complexos pode-se analisar os valores numéricos da matriz, de forma directa ou indirectamente, de modo a encontrar *biclusters* com valores coerentes nas linhas e colunas da matriz, Tabelas 3.2(d) e (e), ou encontrar *biclusters* com evoluções coerentes, Tabelas 3.2(f), (g) e (h). Um *bicluster* com evoluções coerentes pode ser útil se houver interesse em identificar um subconjunto de genes que estejam regulados, acima ou abaixo do normal, num subconjunto de condições sem ter em conta os seus valores de expressão reais ou, se houver interesse, em identificar um subconjunto de condições que têm os mesmos efeitos, ou efeitos opostos, num subconjunto de genes.

Estes são de acordo com as propriedades de cada problema, os *biclusters* geralmente considerados interessantes.

3.3.2 Estrutura dos *biclusters*

Os algoritmos de *biclustering* podem assumir dois tipos de pesquisa. Ou procuram apenas na matriz de dados por um único *bicluster*, Figura 3.4(a), ou então procuram por múltiplos *biclusters* onde o número de *biclusters* que se pretendem encontrar podem, ou não, ser pré-definidos. Como é possível visualizar na Figura 3.4, são várias as estruturas que os *biclusters* podem tomar.

Existem actualmente inúmeros tipos destes algoritmos [2]. Alguns permitem obter *biclusters* sobrepostos, enquanto que outros apenas permitem obter *biclusters* com linhas e(ou) colunas exclusivas. Existem também outro tipo de algoritmos que tentam utilizar todos os elementos da matriz, isto é, tentam fazer com que cada uma das linhas e colunas estejam contidas em pelo menos um *bicluster*. Em dados biológicos, a utilização deste tipo de algoritmos pode não ser muito razoável pois é muito provável que existam linhas e colunas que não pertençam a nenhum *bicluster*. A utilização de algoritmos que obtenham *biclusters* mutuamente exclusivos também não é muito razoável em dados biológicos, visto que, por exemplo, um mesmo gene pode pertencer a mais do que um *bicluster*.

3.3.3 Tipos de algoritmos

Existem um grande número de algoritmos que permitem identificar os mais diversos *biclusters*. Nem todos seguem os mesmos parâmetros para chegar à solução final, alguns apenas permitem identificar um *bicluster* de cada vez ou vários de forma iterativa, outros identificam vários em simultâneo, etc.

Madeira e Oliveira [2] dividem os algoritmos em cinco tipos diferentes:

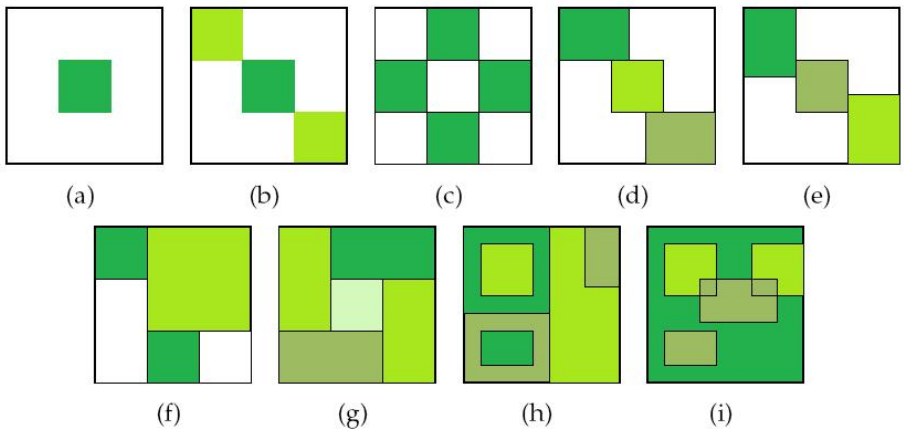

Figura 3.4: Estruturas de *biclusters*: (a) *Bicluster* único; (b) *Biclusters* com linhas e colunas exclusivas; (c) *Biclusters* com estrutura em xadrez; (d) *Biclusters* com linhas exclusivas; (e) *Biclusters* com colunas exclusivas; (f) *Biclusters* não sobrepostos com estrutura em árvore; (g) *Biclusters* não sobrepostos; (h) *Biclusters* sobrepostos com estrutura hierárquica; (i) *Biclusters* sobrepostos e posicionados arbitrariamente.

Combinação iterativa de *clustering* em linhas e colunas: é a forma mais simples e directa de identificar um *bicluster* numa matriz de dados. Consiste em aplicar *clustering* às linhas e colunas separadamente e posteriormente combinar esses resultados através de um processo iterativo.

Divisão e Conquista: segue a abordagem de "dividir para conquistar". Concretamente, quebram o problema em diversos sub-problemas similares ao problema original mas de tamanhos inferiores e resolvem recursivamente esses sub-problemas menores. No final combinam as soluções destes de forma a obterem uma solução para o problema inicial.

Pesquisa *Greedy* (Gulosa): seguem uma heurística gananciosa no sentido em que, a cada passo, é escolhida sempre a opção local óptima na perspectiva que esta leve a uma solução global com qualidade. Normalmente bons *biclusters*, por vezes os melhores, são encontrados com esta técnica. No entanto, pode tomar decisões que levem à perda de boas soluções.

Enumeração exaustiva de *biclusters*: fazem uma procura exaustiva de todos os *biclusters* mediante um conjunto de restrições relativamente ao tamanho destes. Seguem a ideia de que os melhores *biclusters* só podem ser encontrados após a enumeração exaustiva de todas as possibilidades existentes. Têm a desvantagem de, além de em geral serem lentos, necessitarem de restrições quanto ao tamanho dos *biclusters*, em número de linhas e colunas.

Identificação de parâmetros de uma dada distribuição: assume que os *biclusters*

são gerados segundo um determinado modelo estatístico e tenta identificar os parâmetros da distribuição que modela os dados da melhor forma possível.

Capítulo 4

Iterative Signature Algorithm

Neste capítulo serão descritos dois algoritmos de *biclustering*. Por ser o primeiro algoritmo de *biclustering* aplicado a dados de expressão de genes, será feito uma breve introdução ao algoritmo de Cheng e Church[1]. Em seguida, mais detalhadamente, será apresentado o algoritmo que foi utilizado ao longo do trabalho prático nesta presente dissertação. Trata-se de um algoritmo proposto por Bergmann et al[6], sobre matrizes dados de *microarrays*, para identificar módulos de transcrição (*biclusters*) e é conhecido na literatura inglesa por Iterative Signature Algorithm (ISA).

4.1 Algoritmo de Cheng e Church

Cheng e Church [1] foram os primeiros a utilizar o conceito de *biclustering* em dados de expressão de genes de forma a encontrar padrões de co-regulação de genes. O algoritmo desenvolvido por estes encara o problema de *biclustering* como um problema de optimização, definindo um peso (*score*) para cada *bicluster* candidato e desenvolvendo heurísticas que permitam resolver o problema definido pela função peso. Resumidamente, este pode ser considerado um algoritmo *greedy* relaxado que vai forçar a uniformização da matriz dando preferência a submatrizes maiores.

Cheng e Church assumem, implicitamente, que pares (genes e condições) num "bom" *bicluster* têm níveis de expressão constantes, além da possibilidade de, eventualmente, adicionar/remover linhas e colunas específicas. Após remoção de linhas, colunas e das médias das submatrizes, o nível residual deve ser o menor possível. Formalmente, considerando uma matriz X (correspondente à Tabela 3.1) de dados de expressão de genes, um subconjunto de genes $I \subseteq L$ e um subconjunto de condições $J \subseteq C$, defina-se $x_{Ij} = \frac{\sum_{i \in I} x_{ij}}{\#I}$(média do subconjunto de linhas), $x_{iJ} = \frac{\sum_{j \in J} x_{ij}}{\#J}$(média do subconjunto de colunas) e $x_{IJ} = \frac{\sum_{i \in I, j \in J} x_{ij}}{\#I \cdot \#J}$ (média da submatriz). Defina-se também o nível residual (*residue score*) de um elemento x_{ij} numa submatriz X_{IJ} como $RS_{IJ}(i,j) = x_{ij} - x_{Ij} - x_{iJ} + x_{IJ}$ e a função de custo,

mean square residue, como $H(I,J) = \sum_{i\in I, j\in J} \frac{RS_{ij}^2}{\#I \cdot \#J}$, que medirá a similaridade dos *biclusters*. O resultado $H(I,J) = 0$ indicará que os níveis de expressão dos genes flutuam em unissono, isto é, teremos *biclusters* perfeitos com valores coerentes, bem como *biclusters* com valores constantes de variância igual a zero. De forma geral, este algoritmo tem como objectivo encontrar *biclusters* com $H(I,J) \leq \gamma$ ($\gamma \geq 0$) designados γ-*biclusters*. Para atingir este objectivo Cheng e Church[1] propuseram um conjunto de algoritmos *greedy* de adição e remoção de linhas/colunas que combinam de forma a obter um *bicluster*. Caso seja pretendido obter n *biclusters* o algoritmo será repetido n vezes.

No algoritmo 1 está representado em pseudo-código o algoritmo de Cheng e Church para a descoberta de um *bicluster*[9].

Algoritmo 1: Cheng-Church(C, L, E, γ)

Input:

C: condições
L: genes
X: matriz de dados de expressão de genes
γ: média residual máxima de pontuação

begin

1ª Fase - Eliminação(*Deletion*)

Enquanto $(H(I,J) > \gamma)$ fazer:
para $i \in I$ $d(i) = \frac{1}{\#J} \sum_{j\in J} RS_{I,J}(i,j)$.
para $j \in J$ $e(j) = \frac{1}{\#I} \sum_{i\in I} RS_{I,J}(i,j)$.
se $max_{i\in I} d(i) > max_{j\in J} e(j)$ então $I = I \setminus \{argmax_i(d(i))\}$.
senão $J = J \setminus \{argmax_j(e(j))\}$

2ª Fase - Adição(*Addition*)

$I' = I$, $J' = J$
Enquanto$(H(I',J') < \gamma)$ **fazer**:
$I' = I$, $J' = J$
para $i \in C \setminus I$ $d(i) = \frac{1}{\#J} \sum_{j\in J} RS_{I,J}(i,j)$.
para $j \in L \setminus J$ $e(j) = \frac{1}{\#I} \sum_{i\in I} RS_{I,J}(i,j)$.
se $max_{i\in I} d(i) < max_{j\in J} e(j)$ então $I' = I \bigcup \{argmax_i(d(i))\}$.
senão $J' = J \bigcup \{argmax_j(e(j))\}$

end

Output: I, J

O algoritmo 1 pode ser visto como um algoritmo de pesquisa local que começa com todos

os parâmetros da matriz. Isto é, o algoritmo começa por ser inicializado com um *bicluster* que corresponde a todos os elementos da matriz. Dado o parâmetro limiar γ o algoritmo ocorre em duas fases. Numa primeira fase, o algoritmo remove linhas e colunas da matriz original que satisfaçam a condição $H(I, J) > \gamma$. A ideia será remover as linhas/colunas com grande contribuição para o peso melhorando desta forma a média residual do peso total. Na segunda fase do algoritmo, serão adicionadas linhas/colunas, que não aumentem o resíduo actual do *bicluster*.

Após a convergência o algoritmo dará como *output* a submatriz com uma média residual baixa e localmente de tamanho máximo.

4.2 Formalismos

4.2.1 Matriz de observações

Considere-se os dados provenientes de *microarrays* expressos sob a forma de uma matriz X de números reais $n \times m$, onde as linhas representam os genes e as colunas as condições. O elemento da matriz x_{ij} denota a expressão de um gene $i \in L \equiv \{1, \ldots, n\}$ numa determinada condição experimental $j \in C \equiv \{1, \ldots, m\}$, onde n e m representam o número total de genes e condições, respectivamente:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{pmatrix} = [x_{ij}] \quad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, m \end{matrix} \tag{4.1}$$

Seguindo [6], a matriz X poderá ser representado de duas formas possíveis. Como um conjunto de n vectores linha:

$$X = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \tag{4.2}$$

onde cada vector $c_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ (uma linha da matriz) descreve o perfil condição para um gene i. Ou, alternativamente, como um conjunto de m vectores coluna:

$$X = (g_1, g_2, \ldots, g_m) \tag{4.3}$$

onde cada vector $g_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^T$ (uma coluna da matriz) descreve a expressão dos genes para uma condição j.

Em seguida, [6] definem duas matrizes normalizadas da seguinte forma:

$$X_C = \begin{pmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \vdots \\ \hat{c}_n \end{pmatrix} \tag{4.4}$$

e

$$X_L = (\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_m) \tag{4.5}$$

onde as linhas de X_C e as colunas de X_L representam os termos normalizados dos vectores perfis condições e perfis genes, respectivamente, ou seja,

$$\hat{c}_i \equiv \left(\frac{x_{i1} - \bar{x}_{i\cdot}}{s_{i\cdot}}, \ldots, \frac{x_{im} - \bar{x}_{i\cdot}}{s_{i\cdot}} \right) \quad e \quad \hat{g}_j \equiv \left(\frac{x_{1j} - \bar{x}_{\cdot j}}{s_{\cdot j}}, \ldots, \frac{x_{nj} - \bar{x}_{\cdot j}}{s_{\cdot j}} \right)^T \tag{4.6}$$

onde,

$$\overline{x}_{\cdot j} = \textstyle\sum_{i=1}^{n} \frac{x_{ij}}{n} \quad e \quad \overline{x}_{i\cdot} = \textstyle\sum_{j=1}^{m} \frac{x_{ij}}{m}$$

$$s_{\cdot j} = \sqrt{\tfrac{1}{n} \textstyle\sum_{i=1}^{n} (x_{ij} - \overline{x}_{\cdot j})^2} \quad e \quad s_{i\cdot} = \sqrt{\tfrac{1}{m} \textstyle\sum_{j=1}^{m} (x_{ij} - \overline{x}_{i\cdot})^2}$$

Normalizadas as linhas de X_L é então possível que haja comparações entre quaisquer duas condições j e j' através dos perfis $\hat{g}_j$ e $\hat{g}_{j'}$ associados. De modo análogo, o mesmo sucede aquando da normalização de X_C. De notar que, em geral, as matrizes normalizadas X_C e X_L são diferentes.

4.2.2 Módulos de transcrição

Um módulo de transcrição (MT) é um conjunto constituído por um conjunto de genes e um conjunto de condições experimentais tais que esses genes apresentam níveis de expressão elevados sobre as condições experimentais. Um dos objectivos do estudo experimental a apresentar nesta dissertação é, numa primeira fase, encontrar diferentes MT, isto é, vários conjuntos constituídos por um conjunto de genes $L_N \subseteq L$ fortemente correlacionados com um conjunto de condições experimentais $C_N \subseteq C$. Pode-se referir a um MT como um conjunto combinado $M_N = \{L_N, C_N\}$, normalmente também designado de *bicluster*.

Biologicamente, espera-se que um MT possa estar associado em particular a uma função celular, correspondendo cada MT a um factor transcrito que regula determinados genes em L_N e é activado sob determinadas condições em C_N. Esta correspondência entre um factor transcrito (FT) e um MT é uma simplificação do que realmente acontece [6]. Mas esta informação pode vir a tornar-se bastante útil na análise das bases de dados de expressão de genes. Em primeiro lugar porque o número total de factores transcritos, N_{FT},

é muito menor ao número de genes ($N_{FT} \ll n$). Espera-se também que o número de MT, e posteriormente a dimensão da matriz de expressão destes módulos seja relativamente pequena ($N_M \ll n$). Em segundo lugar, o número de genes activados por um simples FT normalmente é bastante limitados. E, por último, diferentes FT podem regular o mesmo gene e, serem activados perante as mesmas condições.

Em termos matemáticos, define-se MT do seguinte modo[6]:

$$\exists (T_C, T_L) : \begin{cases} C_N(L_N) = \{c \in C : (\bar{X}_L)_{g \in L_N} > T_C\}, \\ L_N(C_N) = \{g \in L : (\bar{X}_C)_{c \in C_N} > T_L\} \end{cases} \tag{4.7}$$

onde T_C e T_L são dois parâmetros utilizados como um "filtro" dos níveis de expressão normalizados. A condição (4.7) significa que serão consideradas as condições experimentais c no MT cujo valor médio desses genes, $(\bar{X}_L)_{g \in L_N}$, seja superior a T_C. Analogamente, serão também considerados todos os genes g no MT cujo valor médio do conjunto de condições respectivos, $(\bar{X}_C)_{c \in C_N}$, seja superior a T_L. Esta dependência recíproca entre genes e condições num MT imposta por (4.7) implica que o *bicluster* identificado seja definido por genes com valores de níveis de expressão elevados sob aquelas condições restritas do *bicluster*. Assim, um *bicluster* conterá genes mais correlacionados para as condições do MT e condições mais correlacionadas para os genes do MT, permitindo desta forma uma análise de um conjunto de dados mais coeso e rígido. De notar que a definição apresentada de MT é simétrica no que diz respeito aos genes e condições, isto é, não dá preferência a nenhum deles.

Será agora apresentada uma reformulação baseada em [6] da definição de MT referida em (4.7) através da introdução de notação vectorial. Para isso os genes e as condições presentes no MT denotado por N serão representados por um vector representativo dos genes $g_N = (g_N^1, g_N^2, \ldots, g_N^n)^T$ e um vector representativo das condições $c_N = (c_N^1, c_N^2, \ldots, c_N^m)^T$. Uma componente não nula do vector $g_N(c_N)$ implica que o gene g (condição c) está associada ao módulo N. Considere-se as transformações lineares:

$$c_N^{proj} \equiv X_L^T \cdot g_N = \begin{pmatrix} \hat{g}_1^T g_N \\ \hat{g}_2^T g_N \\ \vdots \\ \hat{g}_m^T g_N \end{pmatrix} \qquad e \qquad g_N^{proj} \equiv X_C \cdot c_N = \begin{pmatrix} \hat{c}_1 c_N \\ \hat{c}_2 c_N \\ \vdots \\ \hat{c}_n c_N \end{pmatrix} \tag{4.8}$$

Os vectores resultantes contêm as projecções dos vectores g_N e c_N, que especificam o MT do conjunto (normalizado) de perfis de genes ($\hat{g}_c$) e perfis de condições ($\hat{c}_g$) da expressão de dados originais definidas em (4.6). Para um vector binário g_N as componentes de c_N^{proj} serão apenas os níveis de expressão somados dos genes do MT para cada condição no conjunto de dados. Assim como para um vector binário c_N os componentes de g_N^{proj} serão apenas os níveis de expressão somados ao longo das condições do MT para cada gene no conjunto de dados.

A consistência requerida em (4.7) pode assim também ser escrita como

$$\exists (T_C, T_L) : \begin{cases} c_N = f_{T_C}(c_N^{proj}), \\ g_N = f_{T_L}(g_N^{proj}) \end{cases} \tag{4.9}$$

onde T_C e T_L são os parâmetros limiares para as condições e genes referidos anteriormente, e f é a função

$$f_t(x) \equiv \begin{pmatrix} w(x_1)\Theta(\tilde{x}_1 - t) \\ \vdots \\ w(x_{N_x})\Theta(\tilde{x}_{N_x} - t) \end{pmatrix} \tag{4.10}$$

que actua separadamente em cada componente x_i dos N_x componentes presentes no vector x, devolvendo o produto de uma função peso $w(x)$ por uma função medida $\Theta(x)$ como *output*. Os argumentos da função de medida, $\tilde{x}_i = \frac{x_i - \mu(x)}{\sigma(x)}$, são centrados utilizando por exemplo a média ($\mu(x) = \bar{x}$), e redimensionados utilizando o desvio padrão, $\sigma(x) = \sqrt{\frac{\sum_i^{N_x}(x_i - \bar{x})^2}{N_x}}$, como um factor de escala. Esta função de medida coloca a zero todos os elementos do vector x que não excedam $\mu(x)$ por pelo menos $t\sigma(x)$. Utilizando $w(x) = 1$ como função de peso todos os elementos significativos são ajustados à unidade. Esta formulação binária corresponde à consistência desejada em (4.7). [6] refere que uma escolha relevante é $w(x) = x$ nos quais $f_t(x)$ é semi-linear.

Em suma, a definição compacta de MT em (4.9) significa que aplicando a função f_{T_C} a c_N^{proj} resulta uma componente $c_N^{(c)}$ do módulo do vector condição c_N diferente de zero se o perfil do gene $\hat{g}_c$ está suficientemente alinhado com o vector gene g_N do módulo. Biologicamente isto significa que um número significativo de genes do módulo estão directamente correlacionados com uma determinado condição c. Analogamente, aplicando a função f_{T_L} a g_N^{proj} resulta uma componente $g_N^{(g)}$ do módulo do vector gene g_N diferente de zero se o perfil da condição $\hat{c}_g$ está suficientemente alinhado com o vector condição c_N do módulo. Biologicamente isto significa que um número significativo de condições do módulo estão directamente correlacionados com um determinado gene g. De realçar, como facilmente se pode concluir, que o conteúdo de um módulo em particular $M_N = \{L_N, C_N\}$ depende do par (T_C, T_L).

4.3 ISA (Iterative Signature Algorithm)

A definição acima descrita de um MT possibilitará a determinação de módulos a partir da matriz de expressão de genes original com base em testes de todos os possíveis conjuntos $\{L_N, C_N\}$ satisfazendo a equação (4.9). No entanto, o número final de conjuntos encontrados cresce exponencialmente com o número de genes e condições, de tal forma que é computacionalmente impraticável a determinação de uma aproximação. Portanto, em seguida, será sugerido uma diferente aproximação segundo [6]. A ideia será procurar soluções da equação de consistência (4.9) utilizando

$$c^{(k+1)} = f_{T_C}(X_L^T \cdot g^{(k)}) \tag{4.11}$$

$$g^{(k+1)} = f_{T_L}(X_C \cdot c^{(k+1)}) \tag{4.12}$$

A primeira equação representa um vector condição $c^{(k+1)}$ para um dado vector gene $g^{(k)}$, onde cada componente deste vector, $c_c^{(k+1)}$, é representativo de um peso (*score*) de uma

condição. Estes *scores* serão não nulos se o correspondente $\hat{g}_c$ definido na equação (4.6) estiver suficientemente alinhado com o vector gene $g_N^{(k)}$. No seguinte passo, na equação (4.12), a componente ou o *score* de um gene, $g_g^{(k+1)}$, do vector gene $g_N^{(k+1)}$, tomará o valor não nulo apenas se a condição $\hat{c}_g$ está suficientemente alinhada com o vector condição $c_N^{(k+1)}$. A estratégia adoptada para cálculo de soluções será reutilizar iterativamente as equações (4.11) e (4.12), e utilizando o vector gene $g^{(1)}$ como *input* nestas obter novos conjuntos de *outputs* $c^{(2)}$ e $g^{(2)}$. Repetindo todo este procedimento obter-se-á $\{g^{(3)}, c^{(3)}\}$ de $g^{(2)}$ e assim sucessivamente até este, ou atingir um ponto de convergência e obter uma solução, ou não atingir um ponto de convergência e atingir um número limite de iterações não conduzindo portanto à obtenção de uma solução.

Em seguida será apresentada uma explicação mais formal de como funciona o ISA no *software* Anaconda aqui utilizado na obtenção de alguns dos resultados apresentados no Capítulo 5.

4.3.1 Descrição da aplicação

Considere-se a matriz X de observações $n \times m$ definida em (4.1). A partir desta obtém-se as duas matrizes normalizadas, uma por colunas, X_L, e outra por linhas, X_C, dadas por:

$$X_L = \left[\frac{x_{ij} - \overline{x}_{.j}}{s_{.j}}\right] \quad \begin{matrix} i : L_i \in L \\ j = 1, \ldots, m \end{matrix} \qquad \text{e} \qquad X_C = \left[\frac{x_{ij} - \overline{x}_{i.}}{s_{i.}}\right] \quad \begin{matrix} i = 1, \ldots, n \\ j : C_j \in C \end{matrix}$$

onde,

$$\overline{x}_{.j} = \sum_{i=1}^{n} \frac{x_{ij}}{n} \quad \text{e} \quad \overline{x}_{i.} = \sum_{j=1}^{m} \frac{x_{ij}}{m}$$

$$s_{.j} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_{.j})^2} \quad \text{e} \quad s_{i.} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(x_{ij} - \overline{x}_{i.})^2}$$

O ISA processa-se essencialmente em duas etapas. Numa primeira etapa para um subconjunto de linhas (escolhidas, ou não, aleatoriamente) são seleccionadas as colunas cujas médias fogem a um padrão definido através de um parâmetro limiar T_C. Na segunda etapa, para o subconjunto das colunas escolhidas na etapa anterior são seleccionadas as linhas cujas médias fogem a um padrão definido através do parâmetro T_L. O ISA tem como objectivo calcular uma sub-matriz da matriz X cujas observações apresentem um comportamento semelhante (em termos de média) tanto ao nível das linhas como das colunas, de acordo com os parâmetros pré-estabelecidos pelo utilizador T_L e T_C.

No algoritmo 2 encontra-se formulado em pseudo-código segundo [7] uma descrição estendida dos passos de uma aplicação do ISA:

Algoritmo 2: ISA

Input: X: matriz de observações $n \times m$

$C = \{C_j, j = 1, \ldots, m\}$ - conjunto dos m valores expressos nas Colunas

$L = \{L_i, i = 1, \ldots, n\}$ - conjunto dos n valores expressos nas Linhas

$L^{(0)}$ - conjunto inicial de $n_0 \leq n$ das linhas escolhidas (não necessariamente de modo aleatório)

X_L e X_C - matrizes normalizadas

T_L - parâmetro limiar associado às linhas

T_C - parâmetro limiar associado às colunas

1ª Fase

Passo 1: Inicializações:

$k = 0$

$[S_{L_i}]_{L_i \in L} = [1 \quad 1 \quad \ldots \quad 1]'$.

Passo 2: Obter a sub-matriz normalizada segundo as linhas seleccionadas $L^{(k)}$, $X_{L^{(k)}}$.

Passo 3: Obter o vector dos scores coluna:

$$[S_{C_1} \quad S_{C_2} \quad \ldots \quad S_{C_m}] = \frac{1}{\#L^{(k)}} [S_{L_i}]'_{L_i \in L^{(k)}} X_{L^{(k)}}.$$

Passo 4: Calcular a média dos scores coluna, $\bar{S}_C = \frac{\sum_j S_{C_j}}{m}$.

Passo 5: Obter o conjunto $C^{(k)}$ de valores expressos C_j das colunas que fogem ao padrão definido em termos do parâmetro T_C do seguinte modo:

$$C^{(k)} = \left\{ C_j \in C : S_{C_j} - \bar{S}_C > T_C \sigma_C \right\} \tag{4.13}$$

onde $\sqrt{\frac{1}{m} \sum_j S_{C_j}^2 - \bar{S}_C^2}$.

2ª Fase

Passo 6: Obter a sub-matriz normalizada segundo as colunas seleccionadas $C^{(k)}$, $X_{C^{(k)}}$.

Passo 7: Actualizar o vector dos scores linha:

$$[S_{L_i}]_{L_i \in L} = \frac{1}{\#C^{(k)}} X_{C^{(k)}} [S_{C_j}]'_{C_j \in C^{(k)}}.$$

Passo 8: Calcular a média dos scores coluna, $\bar{S}_L = \frac{\sum_i S_{L_i}}{n}$.

Passo 9: Obter o conjunto $L^{(k+1)}$ de valores expressos L_i das colunas que fogem ao padrão definido em termos do parâmetro T_L do seguinte modo:

$$L^{(k+1)} = \left\{ L_i \in L : S_{L_i} - \bar{S}_L > T_L \sigma_L \right\} \tag{4.14}$$

onde $\sigma_L = \sqrt{\frac{1}{n} \sum_i S_{L_i}^2 - \bar{S}_L^2}$.

Passo 10: Se $L^{(k+1)} \neq L^{(k)}$ substituir k por $k + 1$ e repetir de Passos 2 a 9.

Passo 11: Se $L^{(k+1)} = L^{(k)}$ STOP.

Output: Um módulo de transcrição $[x_{ij}]_{i \in L^{(k)} j \in C^{(k)}}$

4.3.2 Exemplo

Nota: Tendo em conta os trabalhos dos autores do ISA[6], é aconselhável, de modo geral, utilizar nos Passos 4 e 8 $\bar{S}_C = 0$ e $\bar{S}_L = 0$. No exemplo seguinte será utilizado este aspecto.

Considere-se a seguinte matriz de observações 6×4:

$$\mathbf{X} = \begin{pmatrix} 5 & 0 & 4 & 4 \\ 4 & 2 & 3 & 3 \\ 18 & 17 & 4 & 2 \\ 20 & 15 & 2 & 4 \\ 5 & 1 & 1 & 2 \\ 2 & 3 & 3 & 3 \end{pmatrix} \tag{4.15}$$

Escolham-se as linha 2, 3 e 4 da matriz como conjunto inicial $L^{(0)}$, ou seja, $L^{(0)} = [L_2\ L_3\ L_4]$.

Passo 1:

$k = 0$ e $[S_{L_1}\ S_{L_2}\ S_{L_3}\ S_{L_4}\ S_{L_5}\ S_{L_6}] = [1\ 1\ 1\ 1\ 1\ 1]$

iteração 1

Passo 2:

Calculando $\bar{x}_{.j,j=1,\ldots,6}$ e $s_{.j,j=1,\ldots,6}$ e efectuando as operações necessárias sobre (4.15) obtém-se:

$$\mathbf{X_{L^{(0)}}} = \begin{pmatrix} 0.63706 & -0.57163 & 0.142566 & 0 \\ 1.146706 & 1.407086 & 0.997965 & -1.11803 \\ 1.40153 & 1.143258 & -0.71283 & 1.118034 \end{pmatrix}$$

Passo 3:

$[S_{C_1}\ S_{C_2}\ S_{C_3}\ S_{C_4}] = \frac{1}{3} \cdot [S_{L_2}\ S_{L_3}\ S_{L_4}]' \cdot X_{L^{(0)}} = [0.637059\ 0.659572\ 0.142566\ 0]$

Passo 4:

$\bar{S}_C = 0$

Passo 5:

Seja $T_C = 1$, $\sigma_C = \simeq 0.3383$

Apenas C_1 e C_2 satisfazem a condição (4.13), portanto $C^{(0)} = [C_1\ C_2]$

Passo 6:

Calculando $\bar{x}_{i.,i=1,\ldots,4}$ e $s_{i.,i=1,\ldots,4}$ e efectuando as operações necessárias sobre as colunas seleccionadas $C^{(0)}$ obtém-se:

$$\mathbf{X_{C^{(0)}}} = \begin{pmatrix} 0.789228 & -1.46571 \\ 1.224745 & -1.22474 \\ 0.920296 & 0.801548 \\ 1.126459 & 0.548788 \\ 1.452744 & -0.66034 \\ -1.5 & 0.5 \end{pmatrix}$$

Passo 7:

$[S_{L_1}\ S_{L_2}\ S_{L_3}\ S_{L_4}\ S_{L_5}\ S_{L_6}] = \frac{1}{2} \cdot X_{C^{(0)}} \cdot [S_{C_1}\ S_{C_2}]' = [-0.23198\ -0.01379\ 0.557481\ 0.539793\ -0.3129\ -0.34823]'$

Passo 8:

$\bar{S}_L \simeq 0$

Passo 9:

Considere-se $T_L = 1$ $\sigma_L = 0.377293$
L_3 e L_4 satisfazem a condição (4.14), portanto $L^{(1)} = [L_3\ L_4]$.

Passo 10 ou 11:

Efectuando os cálculos obtem-se que $L^{(1)} \neq L^{(0)}$.

Logo $k = 1$ e nova iteração do algoritmo.

iteração 2

Passo 2:

Calculando $\bar{x}_{.j,j=1,\ldots,6}$ e $s_{.j,j=1,\ldots,6}$ e efectuando as operações necessárias sobre (4.15) obtém-se:

$$\mathbf{X_{L^{(1)}}} = \begin{pmatrix} 1.146706 & 1.407086 & 0.997965 & -1.11803 \\ 1.40153 & 1.143258 & -0.71283 & 1.118034 \end{pmatrix}$$

Passo 3:

$[S_{C_1}\ S_{C_2}\ S_{C_3}\ S_{C_4}] = \frac{1}{3} \cdot [S_{L_2}\ S_{L_3}]' \cdot X_{L^{(1)}} = [0.755357\ 0.758139\ 0.091748\ -0.00929]$

Passo 4:

$\bar{S}_C = 0$

Passo 5:

Seja $T_C = 1$, $\sigma_C = 0.4093$

Apenas C_1 e C_2 satisfazem a condição (4.13), portanto $C^{(0)} = [C_1\ C_2]$

Passo 6:

Calculando $\bar{x}_{i.,i=1,\ldots,4}$ e $s_{i.,i=1,\ldots,4}$ e efectuando as operações necessárias sobre as colunas seleccionadas $C^{(1)}$ obtém-se:

$$\mathbf{X_{C^{(1)}}} = \begin{pmatrix} 0.789228 & -1.46571 \\ 1.224745 & -1.22474 \\ 0.920296 & 0.801548 \\ 1.126459 & 0.548788 \\ 1.452744 & -0.66034 \\ -1.5 & 0.5 \end{pmatrix}$$

Passo 7:

$[S_{L_1}\ S_{L_2}\ S_{L_3}\ S_{L_4}\ S_{L_5}\ S_{L_6}] = \frac{1}{2} \cdot X_{C^{(1)}} \cdot [S_{C_1}\ S_{C_2}]' = [-0.25753\ -0.0017\ 0.651419\ 0.633468\ 0.298357\ -0.37698]'$

Passo 8:

$\bar{S}_L \simeq 0$

Passo 9:

Considere-se $T_L = 1$, $\sigma_L = 0.441273$
L_3 e L_4 satisfazem a condição (4.14), portanto $L^{(2)} = [L_3\ L_4]$.

Passo 10 ou 11:

$L^{(2)} = L^{(1)}$ então o algoritmo pára.

$$Output = \begin{pmatrix} 18 & 17 \\ 20 & 15 \end{pmatrix}$$

Capítulo 5

Estudo Experimental

Neste capítulo, descreve-se o estudo experimental realizado com o objectivo de avaliar e comparar diferentes métodos de selecção de atributos, resultantes da implementação do algoritmo de *biclustering* ISA usando diferentes configurações de parâmetros. Esta avaliação é feita com base no desempenho preditivo de um classificador induzido com os subconjuntos de dados de expressão genética seleccionados. A capacidade preditiva dos modelos de classificação é medida em termos da taxa de erro obtida pelo método de validação cruzada *leave-one-out*.

Os dados de expressão genética induzidos nesse classificador correspondem a apenas determinados subconjuntos de genes. Estes subconjuntos de genes são obtidos através da aplicação de filtros (esquemas de pesagem) criados com base em informação proveniente da combinação de um algoritmo de *biclustering* e do AGA (ver Secção 5.2.2), juntamente com outras técnicas de selecção de atributos (filtros e *wrapper*).

5.1 Bases de dados

Os dados brutos de *microarrays* são influenciados por variações de circunstâncias imprevisíveis que, na sua maioria, dependem de factores técnicos. As experiências de *microarrays* envolvem um número significativo de passos e cada um destes pode introduzir variabilidade nas intensidades medidas afectando a qualidade dos dados. Os métodos de correcção de *background* (CB) e de normalização (NM) são dois métodos de pré-processamento cujo objectivo é refinar os dados brutos com o intuito de reduzir o efeito de variações não desejadas, mas tratando de preservar as variações biológicas intrinsecas aos mesmos.

As bases de dados utilizadas neste trabalho provêem da base de dados brutos Lymphoma[1] (Tabela 5.1), mais concretamente, das bases de dados resultantes do estudo realizado em [17] em que é aplicada, à base de dados referida, um método de correcção de *background*

[1] esta base de dados foi extraída do repositório da Universidade de Stanford [http://genome-www5.stanford.edu/cgi-bin/publication/viewPublication.pl?pub_ no=79]

seguido de um método de normalização. Para este estudo seleccionaram-se 8 bases de dados resultantes de quatro métodos de correcção de *background* (Half, Minimum, Edwards, Normexp) e dois métodos de normalização de dois passos (IgloessSlloess, IlloessSlloess). Todos estes métodos estão descritos pormenorizadamente em [17].

Lymphoma	*Microarrays:108*; # genes 7079; 3 Classes: normal, diffuse large B-cell lymphoma, follicular lymphoma; http://genome-www5.stanford.edu/cgi-bin/publication/viewPublication.pl?pub_ no=79

Tabela 5.1: Informação sobre a base de dados original Lymphoma.

Cada base de dados corresponde a uma matriz de 108 linhas por 7079 colunas, sendo cada linha representativa de um exemplo (um tipo de cancro sobre o qual se conhece o tecido ou amostra de onde provêem os genes, normal, cancro do tipo DLBCL e cancro do tipo FL) e cada coluna representativa de um atributo (neste caso, um gene).

5.2 Métodos de selecção de atributos. Detalhes de implementação

O objectivo desta secção é descrever em pormenor como foram computacionalmente implementadas as diferentes técnicas e métodos de selecção de atributos neste estudo. Assim, a cada conjunto de dados de entrada, resultante da aplicação de um método de correcção de *background* (CB) seguido de um método de normalização (NM), com o objectivo de identificar um subconjunto reduzido de genes altamente discriminativos, é aplicada a seguinte combinação de técnicas de selecção de atributos:

1. Primeiramente é aplicado o algoritmo de *biclustering* ISA (ver Secção 4), ou uma uma variante deste denominado ISA-$Q_{\frac{1}{2}}$ [10], utilizando vários parâmetros de entrada, através de um *software* denominado de Anaconda [30] descrito na Secção 5.2.1. Neste *software*, os dados de entrada foram dispostos numa matriz transposta da matriz de dados original (onde cada gene corresponde a uma linha e cada exemplo a uma coluna). Como produto final, cada aplicação destes algoritmos apresentará um conjunto de *biclusters* com um valor *p-value* associado. Estes parâmetros de saída serão posteriormente utilizados no passo seguinte.

2. A partir do conjunto de *biclusters* obtidos no Anaconda é implementada uma heurística, através de uma aplicação feita à medida em tecnologia Java descrita na Secção 5.2.2, denominada por AGA[2]. Esta permite seleccionar, segundo um critério, os atributos

[2]O nome AGA está associado às pessoas intervenientes na sua construção (**A**ndré Marques, **G**ladys Castillo, **A**delaide Freitas)

(genes) que são considerados mais relevantes para a tarefa de classificação. Aos atributos seleccionados é atribuído um peso de 1 e aos que ficam de fora um peso 0. Como resultado obtém-se um ficheiro com os pesos binários associados a cada gene.

3. A partir do ficheiro de pesos obtido em 2 são filtrados e seleccionados os atributos para continuar no processo que apresentem peso 1. Em seguida, por forma a seleccionar um conjunto de atributos ainda mais restritos, é implementado um esquema de filtro usando MSV. Finalmente, os atributos restantes são usados como dados de entrada num método de selecção *wrapper*. Como resultado obtem-se o melhor subconjunto de genes possível em relação à medida de desempenho seleccionada. Estes métodos foram implementados e avaliados em RapidMiner [29] como descrito na Secção 5.2.3.

5.2.1 Implementação do ISA em Anaconda

Anaconda [28] é um pacote de *software* especialmente desenvolvido por um grupo de Bioinformáticos da Universidade de Aveiro, que junta Informáticos, Estatísticos, Biólogos e Bioquímicos, com o objectivo de investigar a estrutura e propriedades de genomas. Anaconda implementa ferramentas estatísticas para análise de *clustering*, análise de resíduos e histogramas de alguns índices calculados, ferramentas adicionais para mapeamento de sequências de codificações (a fim de detectar, por exemplo, a presença de codões raros), entre outras ferramentas.

O Anaconda foi utilizado ao longo deste trabalho para descoberta de conjuntos de *biclusters* nas bases de dados em estudo. Para isso foi utilizada a ferramenta **BiCluster(ISA)** deste *software*.

ISA no Anaconda

O critério de identificação de *biclusters* no ISA (ver Secção 4.3) é baseado na observação da média[18]. Como é conhecido, a média é uma medida de localização fortemente influenciada por erros ou por pontos isolados que possam aparecer no conjunto de dados. Em dados de *microarrays*, é usual ocorrerem este tipo de situações, podendo assim o ISA identificar eventuais falsos *biclusters* influenciado pela existência de pontos isolados em linha e/ou coluna.

Tendo em conta limitações como esta e baseando-se na estrutura do ISA, em [10] é descrito um novo algoritmo de *biclustering* denominado ISA-$Q_{\frac{1}{2}}$. Este algoritmo tem um modo de funcionamento semelhante ao ISA apresentando no entanto algumas diferenças pontuais: enquanto o ISA trabalha alternativamente com submatrizes de duas matrizes normalizadas (X_L e X_C), o ISA-$Q_{\frac{1}{2}}$ funciona sempre com submatrizes da matriz original X; outra diferença é que no ISA-$Q_{\frac{1}{2}}$ não existem *scores* para as linhas e colunas sendo assim de mais fácil implementação. Em síntese e de modo mais formal, a diferença mais significativa entre um e outro reside nos passos 4 e 8 do algoritmo apresentado na Secção 4.3.1 (Algoritmo 2)

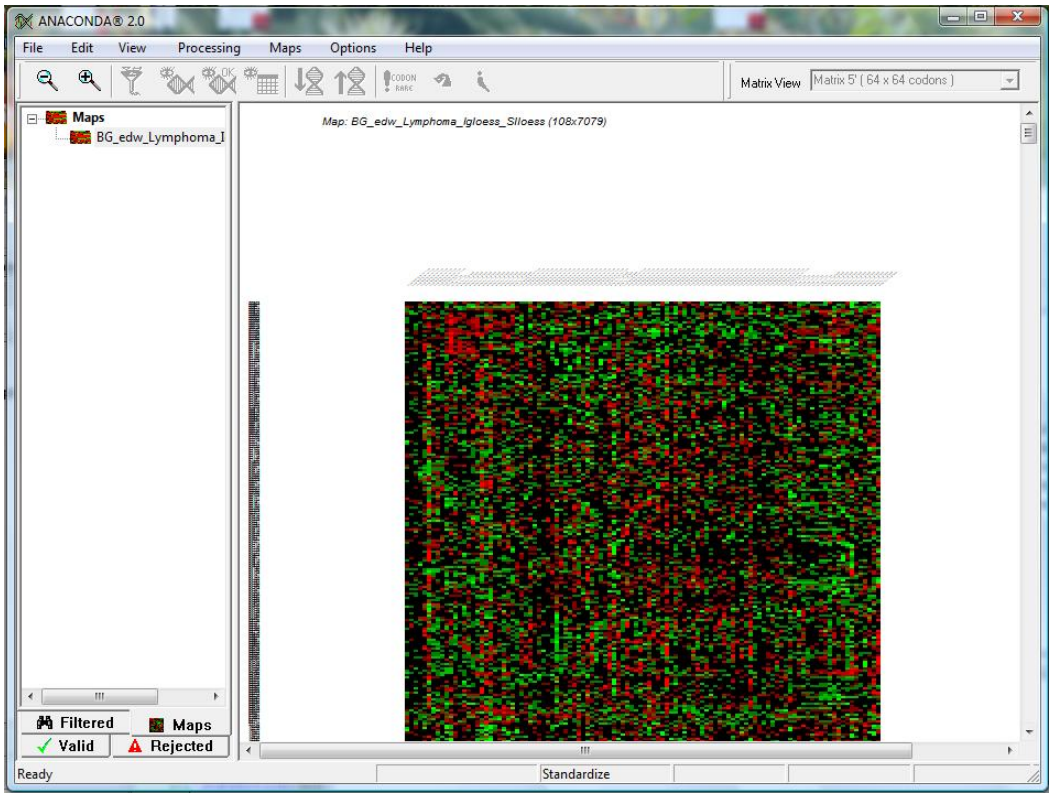

Figura 5.1: Exemplo da visualização de uma matriz de dados de *microarray*, pré-processada segundo um método de correcção de *background* seguindo-se um método de normalização, utilizando o Anaconda. Em coluna estão 108 exemplos (tipos de cancro) e em linha 7079 genes.

onde no ISA é efectuado um cálculo de média enquanto que no ISA-$Q_{\frac{1}{2}}$ será efectuado um cálculo de mediana.

Ambos os algoritmos, ISA e ISA-$Q_{\frac{1}{2}}$, são comparados no presente estudo experimental e ambos estão implementados em Anaconda.

Relativamente ao ISA original foi ainda utilizada uma funcionalidade extra. O ISA foi desenhado originalmente apenas para verificar a condição <X-*Greater*,Y-*Greater*>, onde X representam os exemplos e Y os genes [7, 10]. Estas condições têm como objectivo identificar *biclusters* com valores elevados, em média, por linhas e por colunas, como estabelecem as condições (4.7). No pseudo-código apresentado no Capítulo 4 (Algoritmo 2), estas condições são definidas nos passos 5 (equação (4.13)) e 9 (equação (4.14)). Actualmente na implementação do ISA em Anaconda foram introduzidas outras condições, mais concretamente, <X-*Less*,Y-*Less*> e <X-*Module*,Y-*Module*>, cujos objectivos são semelhantes, com a diferença de que, enquanto na primeira se pretende identificar *biclusters* com os menores valores, na segunda condição o objectivo passa por identificar *biclusters* com valores elevados em módulo. Como ilustrado na interface do Anaconda da Figura

5.2, o utilizador pode optar por utilizar uma das implementações dos algoritmos(ISA ou ISA-$Q_{\frac{1}{2}}$) e combinar todas estas condições para obter diferentes conjuntos de *biclusters*.

Além disso, para cada *bicluster* identificado é ainda associado um *p-value*. Este *p-value* permite identificar se um determinado *bicluster* é ou não denso, isto é, se este é ou não significativo. Resumidamente, ser denso significa que a proporção de elementos na submatriz que define o *bicluster*, com determinada propriedade predefinida em termos de uma discretização, é (significativamente) superior à proporção de elementos com essa propriedade na matriz inicial de dados. O valor utilizado na discretização foi 0.1 sendo este parâmetro introduzido no campo do comando "Discretization" presente na interface do Anaconda (visível no canto inferior direito da Figura 5.2).

Os parâmetros limiares utilizados, T_C e T_L, para a obtenção dos conjuntos de *biclusters* tomaram na maioria dos experimentos o valor 2. No entanto, perante certos casos pontuais em que, ou não foram formados *biclusters* ou, por visualização dos *biclusters* formados no Anaconda (Figura 5.2) com outros valores T_C e T_L, se verificou que estes apresentavam imagens mais homógeneas, isto é, *biclusters* com um padrão de cor pouco diversificado, decidiram-se utilizar também outros valores (para T_C, T_L entre 1 e 1.5).

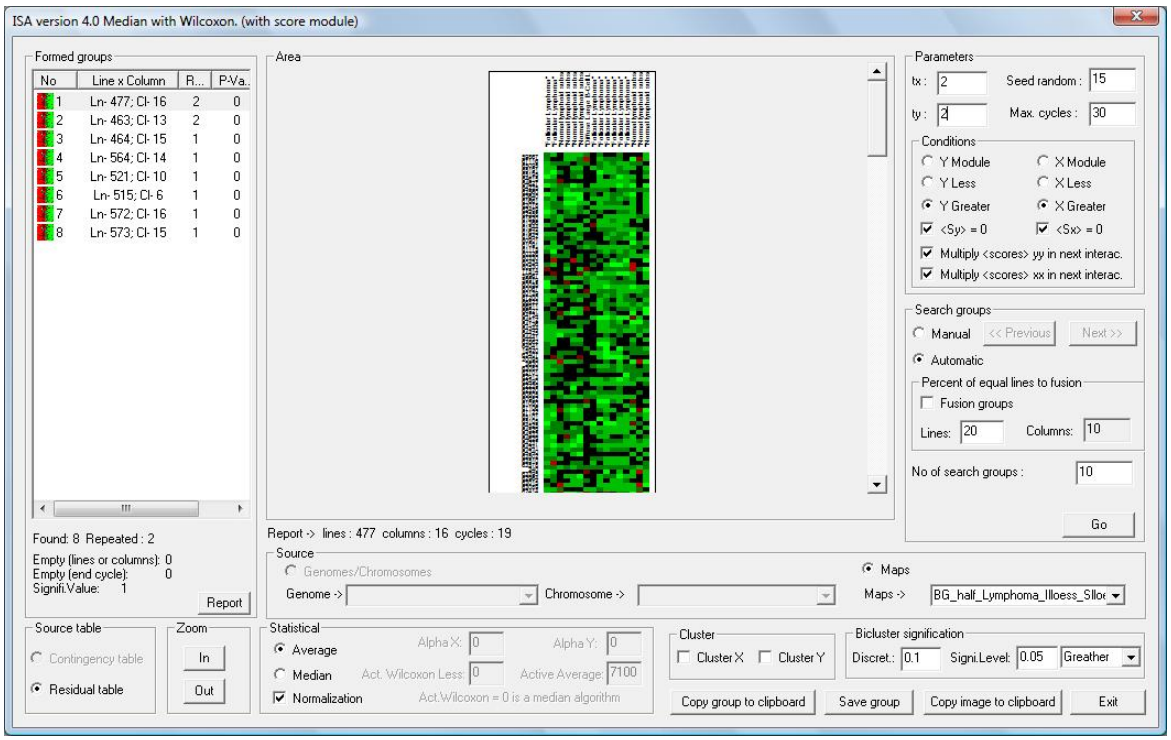

Figura 5.2: Demonstração de funcionalidade relativamente ao ISA no *software* Anaconda.

5.2.2 AGA - uma aplicação para selecção de genes usando *biclusters*

O AGA é uma aplicação Java que implementa uma heurística desenvolvida no contexto deste trabalho (ver pseudo-código no Algoritmo 3), cujo objectivo é seleccionar um conjunto de atributos que melhor discriminam as classes, tomando como entrada o conjunto de *biclusters* resultantes da aplicação do ISA em Anaconda.

Algoritmo 3: Heurística AGA

Input:

C = $\{< B_i, p_i >, i = 1, \ldots, M\}$ - conjunto de *biclusters* e seus respectivos *p-values* obtidos no Anaconda

A = $\{x_1, x_2, ..., x_n\}$ - conjunto de atributos

α - nível de significância

a-value - nível de consistência

begin

Inicializar

$w_j = 0$ para $j = 1, \ldots, n$

Passo 1

$C_A = \{B_i \in C : p_i < \alpha\}$

Passo 2

2.1 Se *supervised selection*

$$C_A = \{B_i \in C_A : \frac{\#\text{classe maioritária}}{\#\text{total de classes}} > \textit{a-value}\}$$

2.2

Para $j = 1$ até n

Se $x_j \in C_A$ então $w_j = 1$

Senão $w_j = 0$

end

Output:

Um vector de pesos binários $W = \{w_1, \ldots, w_n\}$

Após aplicação dos algoritmos de *biclustering* às matrizes de expressão de genes, consoante os parâmetros utilizados, obtem-se um conjunto de *biclusters* e um *p-value* associado a cada *bicluster*. O AGA vai trabalhar sobre esse conjunto com o objectivo de seleccionar genes presentes em *biclusters* que obedeçam a determinadas condições pré-estabelecidas.

AGA permite ao utilizador optar pela implementação de dois mecanismos de filtro sobre o conjunto de *biclusters*: *"unsupervised selection"* e *"supervised selection"* (ver Figura 5.6).

Na opção não supervisionada (*unsupervised selection*), a aplicação faz uma selecção dos atributos (genes) presentes nos *biclusters* sem tomar em consideração a informação da classe. São seleccionados os atributos (genes) dos *biclusters* que apresentem um nível de significância (*p-value*) inferior a um valor α pré-estabelecido (um valor entre 0 e 1). No caso do método supervisionado (*supervised selection*) tem-se em conta a informação sobre a classe de cada exemplo de um *bicluster* sendo pré-definido, além do nível de significância α, um outro parâmetro limiar (um valor entre 0 e 1) denominado *a-value*. Este *a-value* permite seleccionar biclusters *a-consistentes*. Um *bicluster* é *a-consistente* se a proporção de exemplos da classe maioritária é superior ao valor *a-value*. Obviamente, *um bicluster* é 1-consistente se todos os exemplos que o compõem pertencem a uma mesma classe. A ideia básica em que é baseada esta heurística é que caso se consiga obter *biclusters a-consistentes* para *a-values* próximos de 1, então o subconjunto de genes que formam este *bicluster* pode produzir uma boa discriminição da classe maioritária nele representada.

Neste estudo, ao longo dos experimentos, foi usado um nível de significância constante de 0.05 (5%). Em relação ao parâmetro *a-value* definido na opção supervisionada utilizaram-se vários valores neste estudo, mais concretamente, 0.5, 0.8, 0.95 e 1.

Na continuação são apresentadas algumas figuras extraídas da aplicação AGA que ajudam a uma melhor compreensão das funcionalidades aqui expostas.

Formulário inicial

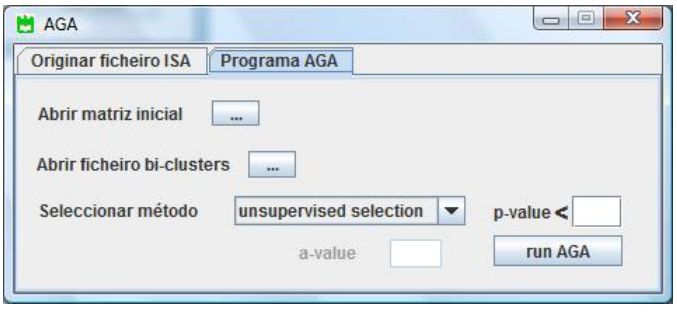

Figura 5.3: Aplicação inicial AGA.

Escolha da matriz de expressão de genes pretendida

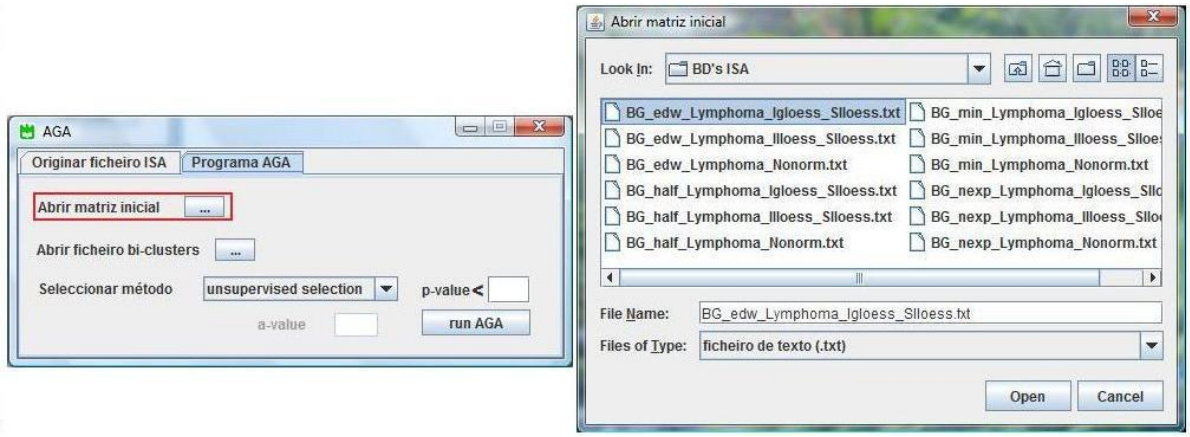

Figura 5.4: Exemplo de selecção do ficheiro da matriz de expressão de genes na aplicação AGA.

Escolha de um conjunto de *biclusters* (provenientes do ISA) associado à matriz anteriormente seleccionada.

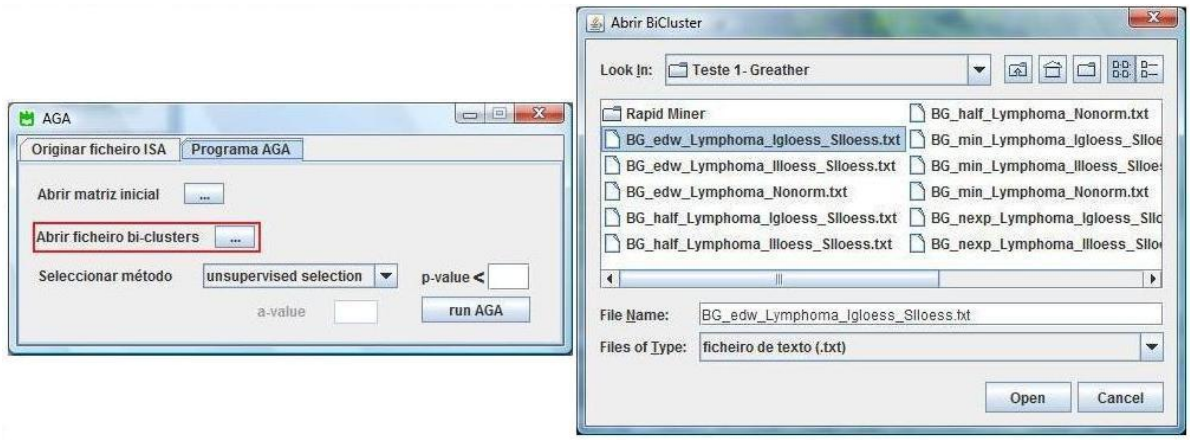

Figura 5.5: Exemplo de selecção do ficheiro dos *biclusters* associado a uma matriz de expressão de genes na aplicação AGA.

Selecção do método pretendido e preenchimento dos respectivos campos necessários

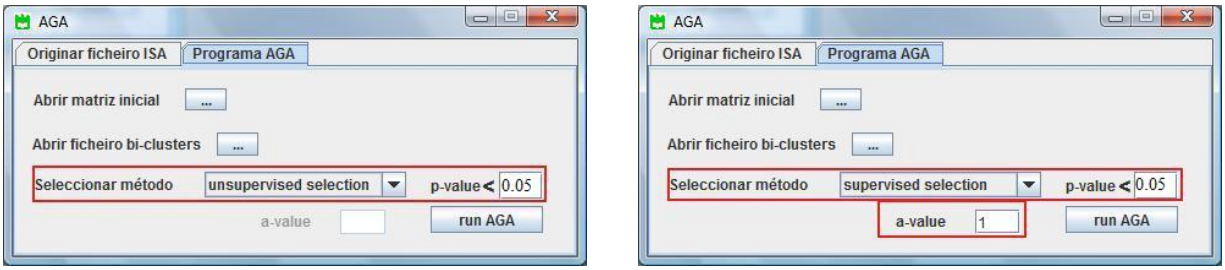

Figura 5.6: Exemplo da escolha dos métodos e preenchimento dos respectivos campos.

Obtenção do ficheiro .wgt com a informação do cada peso associado a cada gene (0 ou 1) correndo a aplicação

```
<?xml version="1.0"
encoding="windows-1252"?>
<attributeweights version="4.3">
    <weight name="59863" value="1"/>
    <weight name="59867" value="0"/>
    <weight name="59871" value="1"/>
    <weight name="59875" value="1"/>
    <weight name="59879" value="1"/>
    <weight name="59883" value="1"/>
    <weight name="59959" value="1"/>
    <weight name="59963" value="1"/>
    <weight name="59966" value="0"/>
    <weight name="59970" value="0"/>
    <weight name="59974" value="1"/>
    <weight name="59978" value="1"/>
```

Figura 5.7: Exemplo de um excerto de um ficheiro *output* do AGA.

5.2.3 Selecção de atributos com o RapidMiner

RapidMiner é uma ferramenta *open-source* especializada em *data mining* e aprendizagem automática. Esta ferramenta permite simular experiências utilizando um grande número de operadores encadeados, descritos em ficheiros XML criados com auxílio da sua interface gráfica. O RapidMiner pode ser usado tanto para investigação como para tarefas de *data mining* em situações reais [29].

O RapidMiner contém mais de meia centena de operadores para os principais procedimentos de aprendizagem automática, tais como operadores para pré-processamento e visualização

de dados. Tem ainda integrados algoritmos de aprendizagem e operadores de selecção de atributos de uma das ferramentas *open-source* mais populares, Weka [33].

No contexto do nosso estudo experimental, o RapidMiner foi usado para continuar com o processo de selecção de atributos a partir da informação obtida com o AGA e também como ferramenta de avaliação do desempenho de classificadores induzidos a partir dos subconjuntos de genes seleccionados. Com esta finalidade, foi criado o projecto em RapidMiner ilustrado ao longo das Figuras 5.8, 5.9 e 5.10. Numa primeira fase é carregada a base de dados de expressão genética que se pretende analisar, assim como o ficheiro de pesos binários (obtido pela aplicação AGA) que indica quais os atributos a serem considerados para futuras análises (apenas atributos com peso 1 serão seleccionados, sendo os restantes eliminados).

Figura 5.8: Fase 1: Selecção de atributos usando pesos obtidos de AGA.

Inicialmente esperava-se que a primeira fase de selecção fosse suficiente para uma redução significativa de atributos, não estando portanto prevista a construção de uma segunda fase. Apesar de, em grande parte, na primeira fase o número de atributos ser reduzido para mais de metade (inicialmente a base de dados continha 7079 atributos), esta selecção revelou-se posteriormente insuficiente na aplicação do algoritmo de aprendizagem. Isto devido ao número de atributos seleccionados continuar elevado (ver Tabela 5.2), produzindo problemas de *out-of-memory* na implementação dos algoritmos *wrapper* de selecção de atributos.

Decidiu-se, portanto, adicionar ao projecto inicial uma *segunda fase* (Figura 5.9) que permitisse reduzir ainda mais o número de atributos provenientes da *primeira fase*. Utilizou-se com esse fim o método de eliminação recursiva de atributos MSV [16]. Para isso são utilizados os operadores SVMWeighting, AttributeWeightSelection e AttributeWeightsApplier. O primeiro operador vai utilizar, internamente, o algoritmo JMySVM[3] para calcular coeficientes para os atributos que serão utilizados como pesos, isto é, são usados os coeficientes de um vector normal de máquinas de suporte vectorial (MSV) lineares para atribuir pesos

[3]O algoritmo JMySVM é uma implementação java do algoritmo mySVM, uma implementação de MSV baseado no algoritmo proposto por Joachims [19]

aos atributos. Este operador trabalha simultaneamente com várias classes (neste caso em concreto três classes). Os operadores AttributeWeightSelection e AttributeWeightsApplier vão seleccionar os atributos que tenham obtido um peso superior a um valor pré-definido.

Figura 5.9: Fase 2: Selecção de atributos usando o esquema de pesagem SVMWeighting.

Por último, utilizou-se um método *wrapper* para seleccionar o melhor subconjunto de atributos em relação a uma medida de bondade (ver Figura 5.10). Apesar de serem computacionalmente mais intensivos, os métodos *wrappers* têm provado ser mais eficientes na obtenção de subconjuntos de atributos mais discriminativos. Como explicado na Secção 2.2.2, estes métodos, durante o processo de procura, usam como medida para um determinado subconjunto de atributos uma estimativa do desempenho da capacidade predictiva de um classificador induzido usando apenas os valores dos atributos seleccionados.

Como classificador utilizou-se o operador LibSVMLearner. Neste operador é utilizado o algoritmo de aprendizagem libsvm da autoria de Chang and Lin[21], que contem vários tipos de MSVs. Para a implementação deste estudo foi escolhido este algoritmo para induzir máquinas de suporte vectorial lineares visto ser um classificador que tem sido usado com muito sucesso, em estudos anteriores, em problemas de classificação de cancro.

Por forma a avaliar o desempenho preditivo dos classificadores induzidos foi usada uma estimativa da taxa de erro obtida através do método de validação cruzada LOO-CV(*left one out cross validation*). Neste método, em cada iteração, um exemplo é deixado de fora para ser testado com o classificador que é induzido utilizando os restantes exemplos. Este procedimento é repetido para todos os exemplos do conjunto de dados. A taxa de erro é então calculada como a proporção dos exemplos incorrectamente classificados do total de exemplos. A principal vantagem do método de validação LOO-CV é que este utiliza o maior número de instâncias possíveis para treino, o que pode aumentar as probabilidades do classificador induzido se tornar mais preciso, principalmente quando se dispõe de poucos exemplos no conjunto de dados. No entanto tem um custo computacional elevado devido ao número de vezes que tem de se treinar um classificador e além disso, é provado que origina uma estimativa do desempenho médio com elevada variância, pelo facto do conjunto de teste em cada iteração apenas ser constituído por um exemplo.

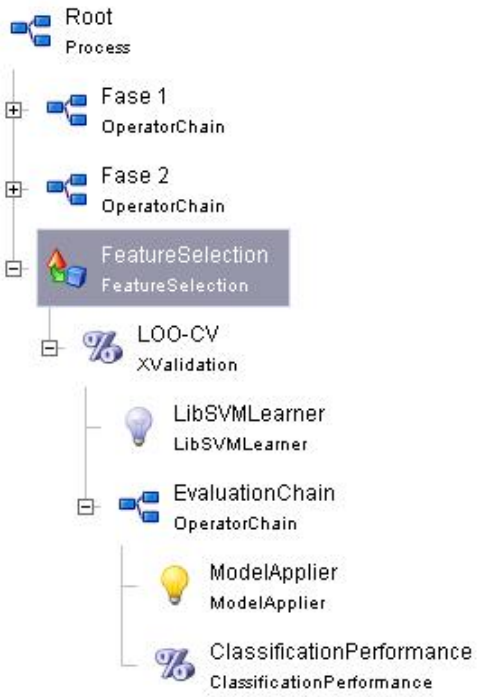

Figura 5.10: Demonstração da Fase 3 utilizada no projecto de RapidMiner.

5.3 Resultados e Análise

5.3.1 Descrição do processo experimental

Na obtenção dos resultados, cada uma das 8 bases de dados, resultantes da combinação de 4 métodos de BC e 2 métodos de NM sobre a base de dados Lymphoma, passou obrigatoriamente pelas três fases anteriormente descritas. Um primeiro passo de construção dos *biclusters* por intermédio dos algoritmos ISA e ISA-$Q_{\frac{1}{2}}$ (Secção 5.2.1) utilizando várias configurações disponíveis. Um segundo de aplicação da heurística AGA desenvolvida (Secção 5.2.2) aplicando diferentes *a-values* (n/a, 0.5, 0.8, 0.95 e 1) e um nível de significância sempre constante ($\alpha = 0.05$). Por fim, um último passo utilizando o RapidMiner (Secção 5.2.3) para, não só seleccionar atributos, como também avaliar o desempenho preditivo de um classificador utilizando os genes seleccionados.

No total foi obtido um conjunto de 30 resultados práticos por cada base de dados ilustrado nas Tabelas 5.4 e 5.5. Estes resultados resultam da combinação da aplicação do ISA e ISA-$Q_{\frac{1}{2}}$, utilizando as várias configurações presentes no Anaconda (<X-*Greater*,Y-*Greater*>, <X-*Less*,Y-*Less*>, <X-*Module*,Y-*Module*>), com a aplicação do AGA utilizando o método "unsupervised selection" e o método "supervised selection" com quatro diferentes *a-values*.

O ISA *Greater*, *Less* e *Module* referidos a seguir correspondem à aplicação ISA utilizando as

condições de <X-*Greater*,Y-*Greater*>, <X-*Less*,Y-*Less*>, <X-*Module*,Y-*Module*> referidas na Secção 5.2.1.

5.3.2 Apresentação e análise de resultados

Analisando a Tabela 5.2 é possível verificar comportamentos distintos entre a selecção de atributos usando o ISA e o ISA-$Q_{\frac{1}{2}}$:

- A diferença entre o número de genes seleccionados como relevantes, usando a heurística AGA, com o método não supervisionado (valor de *a-value*=n/a) quando comparado com o método supervisionado com *biclusters* 0.5-consistentes não é significativa. Aliás aplicando ISA com a condição *Less* apenas existe diferença de um atributo usando o conjunto de dados pré-processado com <BC=min, MN=Il-Sl>. Só usando o ISA-$Q_{\frac{1}{2}}$ com a condição *Greater* o número de genes apresenta ligeiras reduções. Tendo em conta o ISA-$Q_{\frac{1}{2}}$ o número de genes apenas apresenta reduções, ainda que não significativas, na condição *Greater*;

- Por outro lado, à medida que o *a-value* vai aumentando de 0.5 a 1, o que significa que o algoritmo AGA vai seleccionando *biclusters* cada vez mais a-consistentes (com maiores a-values), o ISA só apresenta uma redução mais significativa quando muda de 0.5 para 0.8, mantendo-se quase inalterada para os restantes valores. Pelo contrário, usando o ISA-$Q_{\frac{1}{2}}$, existem vários casos onde a redução é mesmo notável levando à não selecção de nenhum gene.

- O ISA-$Q_{\frac{1}{2}}$ nesta primeira fase de selecção apresenta, salvo certas excepções, uma maior redução do número de genes relativamente ao ISA. Em termos de percentagem esta diferença é bem visível na Tabela 5.3.

Analisando as taxas de erros apresentadas nas Tabelas 5.4 e 5.5, observa-se que utilizando o ISA (Tabela 5.4), em geral, as taxas de erros obtidas evoluem de uma forma constante, para cada condição, aquando da selecção de genes, isto é, as taxas de erro melhoram, pioram ou mantêm-se constantes à medida que se seleccionam *biclusters* mais consistentes (com um a-*value* maior). Existe, no entanto, uma situação pontual que foge à"regra": usando o ISA com a condição *Greater* e a base de dados pré-processada com <BC=min, NM=Il-Sl> como resultado de todo o processo de selecção de genes obtem-se um erro de 0.00%, ou seja, conseguiu-se encontrar um subconjunto de atributos que originam uma taxa de acertos de 100%.

Por outro lado usando o ISA-$Q_{\frac{1}{2}}$ não é possível perceber uma tendência sobre o comportamento da taxa de erro à medida que vão sendo utilizados *biclusters* mais consistentes. Ao contrário do ISA descrito anteriormente, à medida que se vão utilizando *biclusters* mais consistentes os valores das taxas de erro, em cada condição utilizada, com o ISA-$Q_{\frac{1}{2}}$ são, de modo global, inconstantes. Verifica-se também um grande conjunto de condições para os quais não são obtidos resultados práticos. No caso do ISA Module, deve-se ao facto de

Métodos de Pré-Processamento		**a-value**	**ISA**			**ISA-$Q_{\frac{1}{2}}$**		
			Greater	**Less**	**Module**	**Greater**	**Less**	**Module**
BC	**NM**		#genes	#genes	#genes	#genes	#genes	#genes
Edw	**Ig-Sl**	n/a	3116	1765	1673	2322	1611	s.r.a.
		0.5	3116	1765	1673	2277	1611	s.r.a.
		0.8	2689	1748	1423	763	611	s.r.a.
		0.95	2635	1712	1243	591	0	s.r.a.
		1	2635	1712	1243	389	0	s.r.a.
	Il-Sl	n/a	3094	1541	866	1968	1726	s.r.a.
		0.5	3094	1541	866	1948	1726	s.r.a.
		0.8	2881	1541	866	792	585	s.r.a.
		0.95	2881	1484	866	576	0	s.r.a.
		1	2881	1484	866	0	0	s.r.a.
Half	**Ig-Sl**	n/a	1872	1146	774	1434	1785	s.r.a.
		0.5	1872	1146	774	1387	1785	s.r.a.
		0.8	1429	928	416	549	832	s.r.a.
		0.95	1429	768	416	459	337	s.r.a.
		1	1429	768	416	315	337	s.r.a.
	Il-Sl	n/a	1514	1105	1108	1745	1146	s.r.a.
		0.5	1514	1105	1108	1717	1146	s.r.a.
		0.8	1391	1105	1108	606	493	s.r.a.
		0.95	1391	1093	1108	386	0	s.r.a.
		1	1391	1093	1108	0	0	s.r.a.
Min	**Ig-Sl**	n/a	3882	2365	2849	2197	2178	s.r.a.
		0.5	3882	2365	2849	2130	2178	s.r.a.
		0.8	3576	2237	2384	931	801	s.r.a.
		0.95	3566	2226	2384	799	0	s.r.a.
		1	3566	2226	2384	0	0	s.r.a.
	Il-Sl	n/a	3823	2724	1419	2333	2140	s.r.a.
		0.5	3823	2723	1419	2253	2140	s.r.a.
		0.8	3424	2675	1183	874	738	s.r.a.
		0.95	3424	2654	1183	720	0	s.r.a.
		1	3424	2654	1183	566	0	s.r.a.
Nexp	**Ig-Sl**	n/a	4046	3490	3384	3054	2109	s.r.a.
		0.5	4046	3490	3384	2583	2109	s.r.a.
		0.8	3752	3034	2689	1306	766	s.r.a.
		0.95	3752	3023	2689	913	0	s.r.a.
		1	3752	3023	2689	0	0	s.r.a.
	Il-Sl	n/a	4560	3643	4673	3262	1849	s.r.a.
		0.5	4560	3643	4673	3108	1849	s.r.a.
		0.8	4326	3188	3842	1450	756	s.r.a.
		0.95	4297	3103	3685	1054	0	s.r.a.
		1	4297	3103	3685	147	0	s.r.a.

Tabela 5.2: Tabela com os resultados do número de genes resultantes após aplicação dos pesos obtidos de AGA. A abreviatura s.r.a. significa que não se obteram *biclusters* no Anaconda.

	ISA			ISA-$Q_{\frac{1}{2}}$		
a-value	**Greater**	**Less**	**Module**	**Greater**	**Less**	**Module**
n/a	54.25	68.61	70.43	67.66	74.32	s.r.a.
0.5	54.25	68.61	70.43	60.27	74.30	s.r.a.
0.8	58.56	70.94	75.44	87.16	90.14	s.r.a.
0.95	58.72	71.64	76.03	90.29	99.40	s.r.a.
1	58.72	71.64	76.03	97.49	99.40	s.r.a.

Tabela 5.3: Percentagem da redução em média, considerando todas as bases de dados, do número de genes após aplicação dos pesos obtidos de AGA. A abreviatura s.r.a. significa que não se obteram *biclusters* no Anaconda.

não terem sido obtidos *biclusters* no Anaconda, já nos restantes casos está relacionado com o facto de não existirem *biclusters* com o grau de consistência requerido.

A Tabela 5.6 apresenta as taxas de erro para MSV usando o método de avaliação LOO-CV e os subconjuntos de atributos resultantes de um processo de selecção de genes semelhante ao analisado anteriormente, mas sem incluir o passo inicial que implementa o algoritmo de *biclustering*. Isto significa que foi aplicado o processo de selecção de atributos que corresponde à implementação do projecto de RapidMiner da Figura 5.10, mas sem a implementação da "Fase 1", isto é, do primeiro filtro baseado nos pesos resultantes da heurística AGA. Com isto tem-se como objectivo verificar, se de facto o método proposto selecciona subconjuntos de atributos de modo a induzir melhorias nos resultados apresentados, ou se pelo menos não os piora.

Analisando as Tabelas 5.4, 5.5, 5.6 e 5.7 é possivel observar que:

- Com a utilização deste método de selecção de facto existem taxas de erro que melhoram ou se mantém constantes (ver as taxas de erros não negativas da Tabela 5.7);
- Existem no entanto, também uma grande número de situações em que as taxas de erro pioram (ver as taxas de erros negativas da Tabela 5.7);
- O método de selecção que em geral obtém melhores resultados, independentemente do *a-value* tomado, é o ISA-$Q_{\frac{1}{2}}$ *Greater*, e o pior o ISA *Module*;
- Ao contrário do ISA em que existem *biclusters* para qualquer nível de consistência, no ISA-$Q_{\frac{1}{2}}$ esse facto já não se verifica;
- De modo geral, e apesar de em grande parte a diferença não ser muito significativa, com a utilização deste método de selecção obtem-se piores taxas de erro.

Métodos Pré-Processamento		**a-value**	**ISA**					
			Greater		**Less**		**Module**	
CB	**NM**		Erro(%)	#Genes	Erro(%)	#Genes	Erro(%)	#Genes
Edw	**Ig-Sl**	n/a	0.93% ± 9.58%	6	6.48% ± 24.62%	3	12.04% ± 32.54%	4
		0.5	0.93% ± 9.58%	6	6.48% ± 24.62%	3	12.04% ± 32.54%	4
		0.8	0.93% ± 9.58%	3	6.48% ± 24.62%	3	5.56% ± 22.91%	5
		0.95	0.93% ± 9.58%	3	1.85% ± 13.48%	6	5.56% ± 22.91%	5
		1	0.93% ± 9.58%	3	1.85% ± 13.48%	6	5.56% ± 22.91%	5
	Il-Sl	n/a	2.78% ± 16.43%	5	4.63% ± 21.01%	3	5.56% ± 22.91%	5
		0.5	2.78% ± 16.43%	5	4.63% ± 21.01%	3	5.56% ± 22.91%	5
		0.8	1.85% ± 13.48%	6	4.63% ± 21.01%	3	5.56% ± 22.91%	5
		0.95	1.85% ± 13.48%	6	2.78% ± 16.43%	5	5.56% ± 22.91%	5
		1	1.85% ± 13.48%	6	2.78% ± 16.43%	5	5.56% ± 22.91%	5
Half	**Ig-Sl**	n/a	9.26% ± 28.99%	3	0.93% ± 9.58%	6	3.70% ± 18.89%	4
		0.5	9.26% ± 28.99%	3	0.93% ± 9.58%	6	3.70% ± 18.89%	4
		0.8	9.26% ± 28.99%	3	0.93% ± 9.58%	6	12.04% ± 32.54%	2
		0.95	9.26% ± 28.99%	3	0.93% ± 9.58%	6	12.04% ± 32.54%	2
		1	9.26% ± 28.99%	3	0.93% ± 9.58%	6	12.04% ± 32.54%	2
	Il-Sl	n/a	2.78% ± 16.43%	4	0.93% ± 9.58%	5	3.70% ± 18.89%	5
		0.5	2.78% ± 16.43%	4	0.93% ± 9.58%	5	3.70% ± 18.89%	5
		0.8	1.85% ± 13.48%	3	0.93% ± 9.58%	5	3.70% ± 18.89%	5
		0.95	1.85% ± 13.48%	3	9.26% ± 28.99%	2	3.70% ± 18.89%	5
		1	1.85% ± 13.48%	3	9.26% ± 28.99%	2	3.70% ± 18.89%	5
Min	**Ig-Sl**	n/a	3.70% ± 18.89%	4	0.93% ± 9.58%	7	2.78% ± 16.43%	7
		0.5	3.70% ± 18.89%	4	0.93% ± 9.58%	7	2.78% ± 16.43%	7
		0.8	2.78% ± 16.43%	5	0.93% ± 9.58%	7	9.26% ± 28.99%	3
		0.95	2.78% ± 16.43%	5	0.93% ± 9.58%	7	9.26% ± 28.99%	3
		1	2.78% ± 16.43%	5	0.93% ± 9.58%	7	9.26% ± 28.99%	3
	Il-Sl	n/a	0.93% ± 9.58%	5	9.26% ± 28.99%	2	9.26% ± 28.99%	2
		0.5	0.93% ± 9.58%	5	9.26% ± 28.99%	2	9.26% ± 28.99%	2
		0.8	1.85% ± 13.48%	6	9.26% ± 28.99%	2	3.70% ± 18.89%	7
		0.95	1.85% ± 13.48%	6	9.26% ± 28.99%	2	3.70% ± 18.89%	7
		1	1.85% ± 13.48%	6	9.26% ± 28.99%	2	3.70% ± 18.89%	7
Normexp	**Ig-Sl**	n/a	0.93% ± 9.58%	6	1.85% ± 13.48%	5	0.00% ± 0.00%	7
		0.5	0.93% ± 9.58%	6	1.85% ± 13.48%	5	0.00% ± 0.00%	7
		0.8	0.93% ± 9.58%	5	1.85% ± 13.48%	5	1.85% ± 13.48%	5
		0.95	0.93% ± 9.58%	5	1.85% ± 13.48%	5	1.85% ± 13.48%	5
		1	0.93% ± 9.58%	5	1.85% ± 13.48%	5	1.85% ± 13.48%	5
	Il-Sl	n/a	1.85% ± 13.48%	5	1.85% ± 13.48%	5	1.85% ± 13.48%	5
		0.5	1.85% ± 13.48%	5	1.85% ± 13.48%	5	1.85% ± 13.48%	5
		0.8	2.78% ± 16.43%	4	1.85% ± 13.48%	5	4.63% ± 21.01%	4
		0.95	2.78% ± 16.43%	4	1.85% ± 13.48%	5	1.85% ± 13.48%	7
		1	2.78% ± 16.43%	4	1.85% ± 13.48%	5	1.85% ± 13.48%	7

Tabela 5.4: Taxas de erro LOO-CV (%) e número de genes obtidos após aplicação do classificador de MSV (LibSVMLearner). Os genes utilizados nesta classificação resultaram da combinação dos três processos de selecção descritos em 5.2, utilizando no início o algoritmo de *biclustering* ISA.

Métodos Pré-Processamento		a-value	ISA-$Q_{\frac{1}{2}}$					
			Greater		Less		Module	
CB	NM		Erro(%)	#Genes	Erro(%)	#Genes	Erro(%)	#Genes
Edw	Ig-Sl	n/a	0.93% ± 9.58%	6	0.00% ± 0.00%	5	s.r.a.	s.r.a.
		0.5	0.93% ± 9.58%	6	0.00% ± 0.00%	5	s.r.a.	s.r.a.
		0.8	0.00% ± 0.00%	3	3.70% ± 18.89%	8	s.r.a.	s.r.a.
		0.95	9.26% ± 28.99%	3	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		1	0.93% ± 9.58%	3	s.r.RM	s.r.RM	s.r.a.	s.r.a.
	Il-Sl	n/a	1.85% ± 13.48%	3	1.85% ± 13.48%	3	s.r.a.	s.r.a.
		0.5	1.85% ± 13.48%	4	1.85% ± 13.48%	3	s.r.a.	s.r.a.
		0.8	1.85% ± 13.48%	5	9.26% ± 28.99%	3	s.r.a.	s.r.a.
		0.95	10.19% ± 30.25%	2	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		1	s.r.RM	s.r.RM	s.r.RM	s.r.RM	s.r.a.	s.r.a.
Half	Ig-Sl	n/a	2.78% ± 16.43%	4	0.93% ± 9.58%	5	s.r.a.	s.r.a.
		0.5	0.93% ± 9.58%	6	0.93% ± 9.58%	5	s.r.a.	s.r.a.
		0.8	8.33% ± 27.64%	4	6.48% ± 24.62%	6	s.r.a.	s.r.a.
		0.95	8.33% ± 27.64%	4	15.74% ± 36.42%	4	s.r.a.	s.r.a.
		1	4.63% ± 21.01%	4	15.74% ± 36.42%	4	s.r.a.	s.r.a.
	Il-Sl	n/a	2.78% ± 16.43%	4	0.93% ± 9.58%	5	s.r.a.	s.r.a.
		0.5	0.00% ± 0.00%	5	0.93% ± 9.58%	5	s.r.a.	s.r.a.
		0.8	3.70% ± 18.89%	4	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		0.95	9.26% ± 28.99%	2	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		1	s.r.RM	s.r.RM	s.r.RM	s.r.RM	s.r.a.	s.r.a.
Min	Ig-Sl	n/a	1.85% ± 13.48%	4	0.93% ± 9.58%	5	s.r.a.	s.r.a.
		0.5	1.85% ± 13.48%	5	0.93% ± 9.58%	5	s.r.a.	s.r.a.
		0.8	2.78% ± 16.43%	4	7.41% ± 26.19%		s.r.a.	s.r.a.
		0.95	2.78% ± 16.43%	4	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		1	s.r.RM	s.r.RM	s.r.RM	s.r.RM	s.r.a.	s.r.a.
	Il-Sl	n/a	0.93% ± 9.58%	4	0.93% ± 9.58%	4	s.r.a.	s.r.a.
		0.5	2.78% ± 16.43%	3	0.93% ± 9.58%	4	s.r.a.	s.r.a.
		0.8	3.70% ± 18.89%	3	8.33% ± 27.64%	5	s.r.a.	s.r.a.
		0.95	2.78% ± 16.43%	5	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		1	2.78% ± 16.43%	5	s.r.RM	s.r.RM	s.r.a.	s.r.a.
Normexp	Ig-Sl	n/a	1.85% ± 13.48%	6	1.85% ± 13.48%	4	s.r.a.	s.r.a.
		0.5	1.85% ± 13.48%	3	1.85% ± 13.48%	4	s.r.a.	s.r.a.
		0.8	2.78% ± 16.43%	3	10.19% ± 30.25%	4	s.r.a.	s.r.a.
		0.95	0.00% ± 0.00%	7	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		1	s.r.RM	s.r.RM	s.r.RM	s.r.RM	s.r.a.	s.r.a.
	Il-Sl	n/a	1.85% ± 13.48%	6	2.78% ± 16.43%	3	s.r.a.	s.r.a.
		0.5	1.85% ± 13.48%	5	2.78% ± 16.43%	3	s.r.a.	s.r.a.
		0.8	2.78% ± 16.43%	5	4.63% ± 21.01%	6	s.r.a.	s.r.a.
		0.95	2.78% ± 16.43%	5	s.r.RM	s.r.RM	s.r.a.	s.r.a.
		1	2.78% ± 16.43%	2	s.r.RM	s.r.RM	s.r.a.	s.r.a.

Tabela 5.5: Taxas de erro LOO-CV (%) e número de genes obtidos após aplicação do classificador de MSV (LibSVMLearner). Os genes utilizados nesta classificação resultaram da combinação dos três processos de selecção descritos em 5.2, utilizando no início o algoritmo de *biclustering* ISA-$Q_{\frac{1}{2}}$. A abreviatura s.r.a. significa que não se obteram *biclusters* no Anaconda e a abreviatura s.r.RM significa que após aplicação do esquema de pesagem resultante do AGA no RapidMiner não foram seleccionados quaiqueres genes.

CB	**NM**	Erro(%)	#Genes
Edw	**Ig-Sl**	$0.93\% \pm 9.58\%$	3
	Il-Sl	$1.85\% \pm 13.48\%$	5
Half	**Ig-Sl**	$0.00\% \pm 0.00\%$	4
	Il-Sl	$0.93\% \pm 9.58\%$	5
Min	**Ig-Sl**	$2.78\% \pm 16.43\%$	5
	Il-Sl	$0.93\% \pm 9.58\%$	5
Normexp	**Ig-Sl**	$0.00\% \pm 0.00\%$	5
	Il-Sl	$0.93\% \pm 9.58\%$	6

Tabela 5.6: Taxas de erros aplicando o processo de selecção de atributos sem incluir o primeiro filtro baseado nos pesos resultantes da heurística AGA.

Métodos de Pré-Processamento		a-value	ISA			ISA-$Q_{\frac{1}{2}}$		
			Greater	Less	Module	Greater	Less	Module
Edw	Ig-Sl	n/a	0.00	-5.55	-11.07	0.00	0.93	s.r.a.
		0.5	0.00	-5.55	-11.07	0.00	0.93	s.r.a.
		0.8	0.00	-5.55	-4.63	0.93	-2.77	s.r.a.
		0.95	0.00	-0.92	-4.63	-8.33	s.r.RM	s.r.a.
		1	0.00	-0.92	-4.63	-8.33	s.r.RM	s.r.a.
	Il-Sl	n/a	-0.93	-2.78	-11.07	0.00	0.00	s.r.a.
		0.5	-0.93	-2.78	-11.07	0.00	0.00	s.r.a.
		0.8	0.00	-2.78	-4.63	0.00	-7.41	s.r.a.
		0.95	0.00	-0.93	-4.63	-8.25	s.r.RM	s.r.a.
		1	0.00	-0.93	-4.63	s.r.RM	s.r.RM	s.r.a.
Half	Ig-Sl	n/a	-9.26	-0.93	-3.70	-2.78	-0.93	s.r.a.
		0.5	-9.26	-0.93	-3.70	-0.93	-0.93	s.r.a.
		0.8	-9.26	-0.93	-12.00	-8.33	-6.48	s.r.a.
		0.95	-9.26	-0.93	-12.00	-8.33	-15.70	s.r.a.
		1	-9.26	-0.93	-12.00	-4.63	-15.70	s.r.a.
	Il-Sl	n/a	-1.85	0.00	-2.77	-1.85	0.00	s.r.a.
		0.5	-1.85	0.00	-2.77	0.93	0.00	s.r.a.
		0.8	-0.92	0.00	-2.77	-2.77	s.r.RM	s.r.a.
		0.95	-0.92	-8.33	-2.77	-8.33	s.r.RM	s.r.a.
		1	-0.92	-8.33	-2.77	s.r.RM	s.r.RM	s.r.a.
Min	Ig-Sl	n/a	-0.92	1.85	0.00	0.93	1.85	s.r.a.
		0.5	-0.92	1.85	0.00	0.93	1.85	s.r.a.
		0.8	0.00	1.85	-6.48	0.00	-4.63	s.r.a.
		0.95	0.00	1.85	-6.48	0.00	s.r.RM	s.r.a.
		1	0.00	1.85	-6.48	s.r.RM	s.r.RM	s.r.a.
	Il-Sl	n/a	0.00	-8.33	-8.33	0.00	0.00	s.r.a.
		0.5	0.00	-8.33	-8.33	-1.85	0.00	s.r.a.
		0.8	-0.92	-8.33	-2.77	-2.77	-7.40	s.r.a.
		0.95	-0.92	-8.33	-2.77	-1.85	s.r.RM	s.r.a.
		1	-0.92	-8.33	-2.77	-1.85	s.r.RM	s.r.a.
Nexp	Ig-Sl	n/a	-0.90	-1.85	0.00	-1.85	-1.85	s.r.a.
		0.5	-0.90	-1.85	0.00	-1.85	-1.85	s.r.a.
		0.8	-0.93	-1.85	-1.85	-2.78	-10.10	s.r.a.
		0.95	-0.93	-1.85	-1.85	0.00	s.r.RM	s.r.a.
		1	-0.93	-1.85	-1.85	s.r.RM	s.r.RM	s.r.a.
	Il-Sl	n/a	-0.92	-0.92	-0.92	0.00	-1.85	s.r.a.
		0.5	-0.92	-0.92	-0.92	-0.92	-1.85	s.r.a.
		0.8	-1.77	-0.92	-3.70	-0.92	-3.70	s.r.a.
		0.95	-1.77	-0.92	-0.92	-0.92	s.r.RM	s.r.a.
		1	-1.85	-0.92	-0.92	-11.07	s.r.RM	s.r.a.

Tabela 5.7: Taxas de erros resultantes da diferença entre as taxas de erro da Tabela 5.6 e das tabelas 5.4 e 5.5. Os resultados com sinal negativo correspondem a um desempenho preditivo do classificador quando induzido por subconjuntos de genes resultantes do processo de selecção utilizando o primeiro filtro baseado nos pesos resultantes da heurística AGA comparativamente ao método de selecção com ausência deste. A abreviatura s.r.a. significa que não se obteram *biclusters* no Anaconda e a abreviatura s.r.RM significa que após aplicação do esquema de pesagem resultante do AGA no RapidMiner não foram seleccionados quaiqueres genes.

Capítulo 6

Conclusões e trabalhos futuros

Com a importância que a tecnologia de *microarrays* de ADN pode ter nas áreas de biologia molecular e medicina torna-se importante que existam métodos de selecção que permitam efectuar uma selecção de atributos eficaz para os diferentes casos que se possam querer analisar.

Pelos resultados obtidos, e apesar de existirem mais resultados negativos do que positivos, fica a ideia que a selecção de atributos com base apenas nos atributos presentes nos *biclusters* formados pelo ISA pode, de modo geral, obter resultados positivos.

Comparando o ISA-$Q_{\frac{1}{2}}$ com o ISA na selecção de atributos verifica-se que, de modo geral se tivermos apenas em conta as condições para as quais foram obtidos resultados em ambos, ou seja, *Greater* e *Less*, o ISA-$Q_{\frac{1}{2}}$ não só apresenta subconjuntos de genes mais reduzidos, como ainda apresenta resultados mais positivos. Portanto conclui-se que de facto a utilização da mediana em detrimento da média pode ser um melhor método.

Chega-se também à conclusão que, de facto é possivel, utilizando um dos algoritmos de *biclustering* referidos, reduzir no conjunto original um número significativo de genes (66.67% se tivermos em conta o ISA-$Q_{\frac{1}{2}}$ que foi o método que se concluiu apresentar melhores resultados), sem piorar o resultado final de um classificador preditivo. Aliás, é possivel reduzir o conjunto original a um subconjunto de genes, de tal forma relevantes, que melhoram o desempenho do classificador preditivo, melhorando consequentemente as taxas de erros associadas a estes. Este pode ser um factor bastante positivo, visto estar-se a falar de uma redução inicial no conjunto de atributos bastante significativa, e que poderá ser uma mais valia.

Em termos da heurística utilizada as taxas de erro obtidas não seguiram nenhum tipo de padrão em especial, nalguns casos utilizando *biclusters* mais consistentes estas melhoraram ou mantiveram-se constantes, o que é positivo, mas em contra-partida noutras situações pioraram.

Seria interessante num trabalho futuro, com métodos semelhantes aos utilizados neste trabalho, aplicar diferentes algoritmos de *biclustering*, e sobre estes aplicar o AGA, de

modo a comparar as diferenças nos resultados obtidos.

Em relação ao método *wrapper* utilizado já na fase final da selecção de atributos, como referido anteriormente, foi utilizado para todos os testes um classificador de máquinas de suporte vectorial. Outra experiência interessante num trabalho futuro seria utilizar métodos *wrapper*, como neste trabalho, mas com diferentes tipos de classificadores e equiparar as taxas de erros dos resultados obtidos.

Bibliografia

[1] Y. Cheng and G. M. Church, *Biclustering of expression data. In Proc. of 8th International Conference on Intelligent Systems for Molecular Biology*, pages 93-103, 2000.

[2] S. Madeira, A. Oliveira *Biclustering Algorithms for Biological Data Analysis: A Survey*;IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS.

[3] R. Peeters, *The maximum biclique is NP-Complete*, Discrete Applied Mathematics, 131(3):651-654, 2003.

[4] J. Hartigan, *Direct clustering of a data matrix*, Journal of the american Statistical Association, Volume 67, p.123-129, 1972.

[5] M. A. Hall, L. A. Smith, *Feature Selection for Machine Learning Comparing a Correlation based Filter Approach to the Wrapper*, American Association for Artificial Inteligence, 1998.

[6] S. Berman, J. Ihmels, N. Barkai*Iterative signature algorithm for the analysis of large-scale gene expression data*, Phys. Rev., E 67, p. 031902-1-031902-18, 2003.

[7] J. Duarte, A. Freitas, M. Pinheiro, J. L. Oliveira, G. Moura, M. Santos, *ISA: um algoritmo de bi-classificação?*, XIV Congresso Nacional da Sociedade Portuguesa de Estatística, M.E. Ferrão, C.Nunes, C.A. Braumann, 341-349, Covilhã; 2007.

[8] J. Ihmels, G. Friedlander, S. Bergmann, O. Sarig, Y. Ziv., e N. Barkai, *Revealing modular organization in the yeast transcriptional network*, Nat. Genet., 31, p. 370-377, 2002.

[9] A. Tanay, R. Sharan, R. Shamir, *Biclustering Algorithms: A Survey*, 2004.

[10] A. Freitas, M. Pinheiro, V. Freixo, J. Duarte, J. Oliveira, G. Moura, M. Santos, *A median-based Iterative Signature Algorithm*, IASC 2007.

[11] Davidson College, *[Online]: http://gcat.davidson.edu/*

[12] B. Liu, C. Wan, L.Wang, *An Efficient Semi-Supervised Gene Selection Method via SpectralBiclustering*, IEEE TRANSACTIONS ON NANOBIOSCIENCE, VOL. 5, NO. 2, JUNE 2006.

[13] D. P. Berrar, W. Dubitzky, and M. Granzow, *A Practical Approach to Microarray Data Analysis. Norwell*, MA: Kluwer.

[14] Y. Saeys, I. Inza, P. Larrañaga, *A Review of feature selection techniques in bioinformatics, Vol.23 n.19; pages 2507-2517*, 2007.

[15] Ying L. and Jiawei H., *Cancer classification using gene expression data, Information Systems*, Vol.28, pp. 243-268, 2003.

[16] I. Guyon, J. Weston, S. Barnhill, *Gene Selection for Cancer Classification using Support Vector Machines*, Machine Learning, 46, pages 389-422, 2002.

[17] A. Marcos, A. Freitas, G. Castillo, M. Santos, *Avaliação de métodos de correcção de background e normalização em dados de microarrays de ADN complementar*, Actas do XVI Congresso Anual da SPE, 2008.

[18] S. Berman, J. Ihmels, N. Barkai, *Defining transcription modules using large-scale gene expression data*, 2003.

[19] S. Ruping, *mySVM/db - Support Vector Machine for Relational Databases*, online:http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVMDB/index.html

[20] Chih-chung Chang, Chih-jen Lin, *Manuscript Number: 2187 Training v-Support Vector Classifiers: Theory and Algorithms*, 2009.

[21] Chih-chung Chang, Chih-jen Lin, *Libsvm: A library for support vector machines*, 2009.

[22] Samy Bengio, *Statistical Machine Learning from Data - Feature Selection*, 2006.

[23] S. Dudoit, Y. H. Yang, M. J. Callow e T. P. Speed*Statistical methods for identifying differentially expressed genes in replicated cDNA microarrays experiments*, University of California, Berkeley, 2000.

[24] Wentian Li, *Gene Selection for Discriminant Microarray Data Analyses*, Lab of Statistical Genetics Rockfeller University.

[25] . Jaeger, R. Sengupta, W. L. Ruzzo, *Improved gene selection for classification of microarrays*, Pacific Symposium on Biocomputing 8:53-64, 2003.

[26] in Zhou, K. Mao, *LS Bound based gene selection for DNA microarray data*, pages 1559-1564, vol 21 no. 8, 2005.

[27] *Support Vector Machines*, online: http://www.statsoft.com/textbook/stsvm.html#index.

[28] M. Pinheiro, V. Afreixo, G. Moura, A. Freitas, M. A. S Santos, J. L. Oliveira, *Statistical, Computational and Visualization methodologies to unveil gene primary structure features*, Methods Inf Med, 2, 63-168, 2006.

[29] *Site Rapidminer*, online:http://rapid-i.com/component/option,com_frontpage/Itemid,1/lang,en/

[30] *software Anaconda*, online:http://bioinformatics.ua.pt/applications/anaconda

[31] A. Ramalho, *Métodos de Biclustering para a Identificação de Mecanismos de Regulação Genética*, 2006.

[32] S. Busygin, O. Prokopyev, P. M. Pardalos, *Biclustering in data mining*, Volume 35, Pages 2964-2987, Issue 9 (September 2008).

[33] Site visitado: http://www.cs.waikato.ac.nz/ml/weka/

[34] H. Y. Yang, Y. Xiao and M. R. Segal, *Identifying differentially expressed genes from microarray experiments via statistic synthesis*, Bioinformatics, Vol. 21, No. 7, pp.1084-1093, 2004.

[35] Standford Microarray database, online:http://smd.stanford.edu/

[36] Yizong Cheng , George M. Church, *Biclustering of Expression Data*, Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, p.93-103, August 19-23, 2000

[37] H. Cho, I. S. Dhillon, Y. Guan and S. Sra, *Minimum sum-squared residue co-clustering of gene expression data*. In: Proceedings of the fourth SIAM international conference on data mining

[38] Q. Sheng, Y. Moreau and B. Moor, *Biclustering microarray data by Gibbs sampling*. Bioinformatics. v19. ii196-ii205.

[39] Ron Kohavi and George John, *Wrappers for Feature Subset Selection (late draft)*, In Artificial Intelligence journal, special issue on relevance, Vol. 97, Nos 1-2, pp. 273-324.NEC's

[40] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, Z. Yakhini, *Tissue Classification with Gene Expression Profiles*, Journal of Computational Biology

[41] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, E. S. Lander, *Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring*, Vol. 286. no. 5439, pp. 531 - 537, Science 15 October 1999

Anexos

Anexo 1 - Código Java da aplicação AGA

Classe do algoritmo AGA e auxiliares

```
package aga;

import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class AlgoritmoAGA {

  private String path1; //correspondente ao path da matriz original
  private String path2; //correspondente ao path do conjunto dos bi-clusters
  private String pathOutput;
  private double pvalue;
  BufferedReader f = null;

  /**
   * Construtor
   * @param path1
   * @param path2
   */
  public AlgoritmoAGA(String path1, String path2, String path3, double
      pvalue){
    this.path1 = path1;
    this.path2 = path2;
    this.pathOutput = path3;
    this.pvalue = pvalue;
  }

  /**
   * Retorna um arrayList com todas as classes da matriz original
   * @return
   * @throws IOException
   */
  private List<String> openMatriz() throws IOException{
```

```
    List<String> L1 =  new ArrayList<String>();
    String s;

    //Abrir o ficheiro correspondente à matriz original
    try{
      f = new BufferedReader(new FileReader(this.path1));
    }catch(Exception e){
      return null;
    }

    //Ler a primeira linha do ficheiro, pois é aí que se encontram todas as
       classes da matriz original
    s = f.readLine();
    int pos1 = s.indexOf("\t")+1;
    int pos2 = s.indexOf("\t", pos1) ;

    //guardar cada classe que foi lida na 1ªlinha num arrayList
    while(pos2>0&&(pos1!=pos2)){  //Enquanto não houver mais tabs ou apenas
       tabs
      L1.add(s.substring(pos1, pos2));
      pos1 = pos2+1;
      pos2 = s.indexOf("\t", pos1);
      if (pos2==-1)
          L1.add(s.substring(pos1));
    }
    f.close();
    return L1;
  }

  /**
   * Retorna true se um bicluster é consistente, caso contrário retorna
      false
   * @return
   */
  private boolean consistente(List<Objecto> L, double grauConsistencia){
    double max=0;
    double total=0;
    for(int i=0; i<L.size();i++){
      if (L.get(i).numOcurrencias>max)
        max=L.get(i).numOcurrencias;
    total += L.get(i).numOcurrencias;
    }
    return ((max/total)>=grauConsistencia);
  }

  /**
   * Retorna um ficheiro xml com a informação dos atributos full-
      consistentes ou a-consistentes
   * @param L
   * @param c
   * @return
```

```
     * @throws IOException
     */
    private void imprimir(List<Integer> L) throws IOException{
      Collections.sort(L);  //ordenar a lista
      int aux = 0, k=0;

      //Abrir o ficheiro correspondente à matriz original
      try{
        f = new BufferedReader(new FileReader(this.path1));
      }catch(Exception e){
      }

      f.readLine(); //Ignorar a 1ªlinha

      BufferedWriter g = null;
      try {
        // abrir o ficheiro de texto
        g = new BufferedWriter(new FileWriter(this.pathOutput));
      } catch (Exception e) {}

      g.write("<?xml version=\"1.0\" encoding=\"windows-1252\"?>\n<
          attributeweights version=\"4.3\">");

      String s = f.readLine();

      while (s!=null){
        if((k==L.size())||(aux != L.get(k))){
          g.write("\n");
          g.write("    <weight name=\""+s.substring(0, s.indexOf("\t"))+"\"
              value=\"0\"/>");
        }
        else{
          g.write("\n");
          g.write("    <weight name=\""+s.substring(0, s.indexOf("\t"))+"\"
              value=\"1\"/>");
          k++;
        }
        aux++;
        s = f.readLine();
      }

      g.write("\n</attributeweights>");
      g.close();
      f.close();
    }

    /**
     * Algoritmo Principal
     * @param a - referente à consistência que o utilizador pretende
     * @return
     * @throws IOException
     */
```

```
public void algoritmoAGA(double a) throws IOException{
  List<String> L1 = this.openMatriz();  //arrayList com as classes do
     ficheiro original
  List<String> L2 = new ArrayList<String>();  //arrayList auxiliar que
     guarda as classes de um determinado bicluster
  List<Objecto> L3 = new ArrayList<Objecto>();  //arrayList auxiliar que
     guarda as classes de um determinado bicluster assim como as suas
     ocurrências
  List<Integer> ListaIndividuos = new ArrayList<Integer>(); //arrayList
     que vai armazenar os indices das linhas onde estão os individuos
     pertencentes aos bi-clusters a-consistentes

  //Abrir o ficheiro do output com a informação dos biclusters
  try{
    f = new BufferedReader(new FileReader(this.path2));
  }catch(Exception e){}

  String pval = f.readLine();
  //Passar logo à 2ªlinha uma vez que é a partir desta que está a
     informação que interessa
  while(pval!=null){
    pval = pval.substring(pval.indexOf("p-value:")+8).trim().replace(",",
       ".");//retira o valor do p-value do bicluster
    if (Double.parseDouble(pval)<=this.pvalue){ //Se o p-value desse bi-
       cluster for menor ou igual ao inserido pelo utilizador faz todo o
       processo, caso contrário passa para o próximo
      String ind = f.readLine();
      int pos1 = 0;
      int pos2 = ind.indexOf("\t") ;

      while ((pos2>0)&&(pos1!=pos2)){ //enquanto não chegar ao fim da
         linha ou só encontrar tabs nessa linha
        L2.add(L1.get(Integer.parseInt((ind.substring(pos1, pos2).trim()))
           ));
        pos1 = pos2+1;
        pos2 = ind.indexOf("\t", pos1);

        if (pos2==-1)
          try{
            L2.add(L1.get(Integer.parseInt((ind.substring(pos1).trim()))))
               ;
          }catch (Exception b){}
      }

      //Só vai considerar biclusters com 3 colunas ou mais
      if (L2.size()>2){
        //armazena num arrayList os nomes das classes com a informação do
           numero de ocurrências
        while(!(L2.isEmpty())){
          Objecto obj = new Objecto();
          obj.nomeClasse = L2.get(0);
          int contador = 0;
```

```
            while (L2.contains(obj.nomeClasse)){
              L2.remove(obj.nomeClasse);
              contador++;
            }
            obj.numOcurrencias = contador;
            L3.add(obj);
            obj=null;
          }

          //verificar se é a-consistente e caso seja guardar na arrayList
             para bi-clusters a-consistentes
          if(this.consistente(L3, a)){
            String individuos = f.readLine(); //String que guarda a linha do
                ficheiro de output do ISA que tem a informação relativamente
                aos atributos do bi-cluster que se está a analisar
            pos1 = 0;
            pos2 = individuos.indexOf("\t") ;
            while ((pos2>0)&&(pos1!=pos2)){
              if(!ListaIndividuos.contains(Integer.parseInt(individuos.
                  substring(pos1, pos2).trim())))
                ListaIndividuos.add(Integer.parseInt(individuos.substring(
                    pos1, pos2).trim()));
              pos1 = pos2+1;
              pos2 = individuos.indexOf("\t", pos1);
              if (pos2==-1){
                try{
                  if (!ListaIndividuos.contains(Integer.parseInt(individuos.
                      substring(pos1).trim())))
                    ListaIndividuos.add(Integer.parseInt(individuos.
                        substring(pos1).trim()));
                }catch (Exception f){};
              }
            }
            individuos="";
          }else f.readLine(); //Apenas passa uma linha à frente no ficheiro
        }
        else{
          L2.clear();
          f.readLine();
        }
      }
      else{
        f.readLine();f.readLine();
      }
      L3.clear(); //Limpar a lista
      f.readLine();
      pval = f.readLine();
    }
    //fechar o ficheiro
    f.close();

    this.imprimir(ListaIndividuos);
```

```
  }

  /**
   * Guarda numa lista todos os atributos referentes ao output do ISA
   */
  public void listaISA() throws IOException{
    List<Integer> L = new ArrayList<Integer>(); //arrayList que vai
        armazenar os indices dos atributos de output do ISA

    //Abrir o ficheiro do output com a informação dos biclusters
    try{
      f = new BufferedReader(new FileReader(this.path2));
    }catch(Exception e){}

    f.readLine();

    String colunas = f.readLine();

    String ind = f.readLine();

    while(ind != null){
      String[] numColunas = colunas.split("\t");
      if (numColunas.length>2){
        int pos1 = 0;
        int pos2 = ind.indexOf("\t") ;
        while ((pos2>0)&&(pos1!=pos2)){ //enquanto não chegar ao fim da
            linha ou só encontrar tabs nessa linha
          if (!L.contains(Integer.parseInt(ind.substring(pos1, pos2))))
            L.add(Integer.parseInt(ind.substring(pos1, pos2)));
          pos1 = pos2+1;
          pos2 = ind.indexOf("\t", pos1);

          if (pos2==-1)
            try{
              if (!L.contains(Integer.parseInt(ind.substring(pos1))))
              L.add(Integer.parseInt(ind.substring(pos1)));
            }catch (Exception b){}
        }
      }
      f.readLine();f.readLine();
      colunas = f.readLine();
      ind = f.readLine();
    }
    f.close();

    this.imprimir(L);
  }
}
```

Classe da Interface

```
package aga;

import java.io.*;
import javax.swing.*;
import java.awt.Rectangle;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import java.awt.Dimension;
import java.awt.Point;
import javax.swing.JTabbedPane;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.DefaultComboBoxModel;
import java.lang.String;
import java.awt.Toolkit;
import java.awt.Font;

public class Interface {

  private JFrame jFrame = null;  //  @jve:decl-index=0:visual-constraint
     ="17,13"
  private JPanel jContentPane1 = null;
  private JFileChooser jc;  //para o OpenFileDialog
  private String path1; //string para armazenar o path da matriz original
  private String path2; //string para armazenar o path do conjunto dos
     biclusters
  private String path3; //string para armazenar o path de uma matriz para
     transformar
  private String path4; //string para armazenar o path do ficheiro XML da
     matriz a transformar
  private String pathFicheiroGuardar;   //string para armazenar o path do
     ficheiro transformado necessário para o ISA
  private String pathOutputXML;   //string para armazenar o path do ficheiro
      de Output do AGA
  private JLabel jLabelGravarMatriz = null;
  private JLabel jLabelFicheiro_a_Transformar = null;
  private JButton jButtonFichTransformar = null;
  private JLabel jLabelFicheiroXML = null;
  private JButton jButtonFichXML = null;
  private JButton jButtonTransformar = null;
  private JTabbedPane jTabbedPane = null;
  private JPanel jContentPane2 = null;
  private JLabel jLabelOpenMatriz1 = null;
  private JButton jButtonAbrirMatriz1 = null;
  private JLabel jLabelAbrirBiclusters1 = null;
  private JButton jButtonAbrirBiclusters1 = null;
  private JComboBox jComboBoxMetodo1 = null;
  private JLabel jLabelMetodo1 = null;
```

```
private JButton jButtonAGA1 = null;
private JLabel jLabel_aValue1 = null;
private JTextField jTextField_aValue1 = null;
private JLabel jLabelpvalue = null;
private JTextField jTextField_pvalue = null;
private JLabel jLabelSinalMenor = null;

/**
 * This method initializes jButtonFichTransformar
 *
 * @return javax.swing.JButton
 */
private JButton getJButtonFichTransformar() {
  if (jButtonFichTransformar == null) {
    jButtonFichTransformar = new JButton();
    jButtonFichTransformar.setText("...");
    jButtonFichTransformar.setLocation(new Point(141, 56));
    jButtonFichTransformar.setSize(new Dimension(34, 17));
    jButtonFichTransformar.addActionListener(new java.awt.event.
        ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent e) {
        jc = new JFileChooser(path3);
        ExampleFileFilter filtro = new ExampleFileFilter();
        filtro.addExtension("dat");
        filtro.setDescription("base dados");
        jc.setFileFilter(filtro);
        jc.setDialogTitle("Abrir ficheiro original");
        jc.showOpenDialog(null);
        //Obter o path da matriz original
        File file = jc.getSelectedFile();
        if (file!=null){
          path3 = file.getPath();
        }
      }
    });
  }
  return jButtonFichTransformar;
}

/**
 * This method initializes jButtonFichXML
 *
 * @return javax.swing.JButton
 */
private JButton getJButtonFichXML() {
  if (jButtonFichXML == null) {
    jButtonFichXML = new JButton();
    jButtonFichXML.setBounds(new Rectangle(141, 86, 34, 19));
    jButtonFichXML.setPreferredSize(new Dimension(43, 26));
    jButtonFichXML.setText("...");
    jButtonFichXML.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent e) {
```

```
            jc = new JFileChooser(path4);
            ExampleFileFilter filtro = new ExampleFileFilter();
            filtro.addExtension("aml");
            filtro.setDescription("ficheiro XML");
            jc.setDialogTitle("Abrir ficheiro XML");
            jc.setFileFilter(filtro);
            jc.showOpenDialog(null);
            //Obter o path da matriz original
            File file = jc.getSelectedFile();
            if (file!=null){
              path4 = file.getPath();
            }
          }
        });
    }
    return jButtonFichXML;
}

/**
 * This method initializes jButtonTransformar
 *
 * @return javax.swing.JButton
 */
private JButton getJButtonTransformar() {
    if (jButtonTransformar == null) {
      jButtonTransformar = new JButton();
      jButtonTransformar.setBounds(new Rectangle(291, 90, 149, 22));
      jButtonTransformar.setText("Originar ficheiro");
      jButtonTransformar.addActionListener(new java.awt.event.ActionListener
          () {
        public void actionPerformed(java.awt.event.ActionEvent e) {
          jc = new JFileChooser(pathFicheiroGuardar);
          ExampleFileFilter filtro = new ExampleFileFilter();
          filtro.addExtension("txt");
          filtro.setDescription("ficheiro ISA");
          jc.setFileFilter(filtro);
          jc.setDialogTitle("Gravar ficheiro transformado");
          jc.setSelectedFile(new File(path3.substring(path3.lastIndexOf("\\"
              )+1, path3.lastIndexOf(".dat"))+".txt"));
          jc.showSaveDialog(null);
          File file = jc.getSelectedFile();
          if (file!=null){
            pathFicheiroGuardar = file.getPath();
          }
          try{
            TransformarMatriz a = new TransformarMatriz(path3, path4,
                pathFicheiroGuardar);
            a.Transformar();
          }catch (IOException a){};
        }
      });
    }
```

```
    return jButtonTransformar;
  }

  /**
   * This method initializes jTabbedPane
   *
   * @return javax.swing.JTabbedPane
   */
  private JTabbedPane getJTabbedPane() {
    if (jTabbedPane == null) {
      jTabbedPane = new JTabbedPane();
      jTabbedPane.addTab("Originar ficheiro ISA", null, getJContentPane1(),
          null);
      jTabbedPane.addTab("Programa AGA", null, getJContentPane2(), null);
    }
    return jTabbedPane;
  }

  /**
   * This method initializes jContentPane2
   *
   * @return javax.swing.JPanel
   */
  private JPanel getJContentPane2() {
    if (jContentPane2 == null) {
      jLabelSinalMenor = new JLabel();
      jLabelSinalMenor.setBounds(new Rectangle(385, 83, 16, 22));
      jLabelSinalMenor.setFont(new Font("Dialog", Font.BOLD, 18));
      jLabelSinalMenor.setText("<");
      jLabelpvalue = new JLabel();
      jLabelpvalue.setText("p-value");
      jLabelpvalue.setSize(new Dimension(47, 22));
      jLabelpvalue.setLocation(new Point(340, 83));
      jLabel_aValue1 = new JLabel();
      jLabel_aValue1.setText("a-value");
      jLabel_aValue1.setSize(new Dimension(58, 22));
      jLabel_aValue1.setLocation(new Point(201, 112));
      jLabel_aValue1.setEnabled(false);
      jLabelMetodo1 = new JLabel();
      jLabelMetodo1.setBounds(new Rectangle(15, 81, 128, 24));
      jLabelMetodo1.setText("Seleccionar método");
      jLabelAbrirBiclusters1 = new JLabel();
      jLabelAbrirBiclusters1.setBounds(new Rectangle(14, 48, 142, 24));
      jLabelAbrirBiclusters1.setText("Abrir ficheiro bi-clusters");
      jLabelOpenMatriz1 = new JLabel();
      jLabelOpenMatriz1.setBounds(new Rectangle(15, 12, 113, 24));
      jLabelOpenMatriz1.setEnabled(true);
      jLabelOpenMatriz1.setText("Abrir matriz inicial");
      jContentPane2 = new JPanel();
      jContentPane2.setLayout(null);
      jContentPane2.add(jLabelOpenMatriz1, null);
      jContentPane2.add(getJButtonAbrirMatriz1(), null);
```

```
        jContentPane2.add(jLabelAbrirBiclusters1, null);
        jContentPane2.add(getJButtonAbrirBiclusters1(), null);
        jContentPane2.add(getJComboBoxMetodo1(), null);
        jContentPane2.add(jLabelMetodo1, null);
        jContentPane2.add(getJButtonAGA1(), null);
        jContentPane2.add(jLabel_aValue1, null);
        jContentPane2.add(getJTextField_aValue1(), null);
        jContentPane2.add(jLabelpvalue, null);
        jContentPane2.add(getJTextField_pvalue(), null);
        jContentPane2.add(jLabelSinalMenor, null);
    }
    return jContentPane2;
}

/**
 * This method initializes jButtonAbrirMatriz1
 *
 * @return javax.swing.JButton
 */
private JButton getJButtonAbrirMatriz1() {
    if (jButtonAbrirMatriz1 == null) {
        jButtonAbrirMatriz1 = new JButton();
        jButtonAbrirMatriz1.setBounds(new Rectangle(139, 15, 34, 18));
        jButtonAbrirMatriz1.setEnabled(true);
        jButtonAbrirMatriz1.setText("...");
        jButtonAbrirMatriz1.addActionListener(new java.awt.event.
            ActionListener() {
          public void actionPerformed(java.awt.event.ActionEvent e) {
            jc = new JFileChooser(path1);
            ExampleFileFilter filtro = new ExampleFileFilter();
            filtro.addExtension("txt");
            filtro.setDescription("ficheiro de texto");
            jc.setFileFilter(filtro);
            jc.setDialogTitle("Abrir matriz inicial");
            jc.showOpenDialog(null);
            //Obter o path da matriz original
            File file = jc.getSelectedFile();
            if (file!=null){
              path1 = file.getPath();
            }
          }
        });
    }
    return jButtonAbrirMatriz1;
}

/**
 * This method initializes jButtonAbrirBiclusters1
 *
 * @return javax.swing.JButton
 */
private JButton getJButtonAbrirBiclusters1() {
```

```
    if (jButtonAbrirBiclusters1 == null) {
      jButtonAbrirBiclusters1 = new JButton();
      jButtonAbrirBiclusters1.setBounds(new Rectangle(165, 51, 34, 19));
      jButtonAbrirBiclusters1.setText("...");
      jButtonAbrirBiclusters1.addActionListener(new java.awt.event.
          ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent e) {
          jc = new JFileChooser(path2);
          ExampleFileFilter filtro = new ExampleFileFilter();
          filtro.addExtension("txt");
          filtro.setDescription("ficheiro de texto");
          jc.setFileFilter(filtro);
          jc.setDialogTitle("Abrir BiCluster");
          jc.showOpenDialog(null);
          //Obter o path da matriz original caso possivel
          File file = jc.getSelectedFile();
          if (file!=null){
            path2 = file.getPath();
          }
        }
      });
    }
    return jButtonAbrirBiclusters1;
  }

  /**
   * This method initializes jComboBoxMetodo1
   *
   * @return javax.swing.JComboBox
   */
  private JComboBox getJComboBoxMetodo1() {
    if (jComboBoxMetodo1 == null) {
      jComboBoxMetodo1 = new JComboBox();
      jComboBoxMetodo1.setBounds(new Rectangle(155, 83, 165, 21));
      jComboBoxMetodo1.setModel(new DefaultComboBoxModel(new String[] {"
          unsupervised selection", "supervised selection"}));
      jComboBoxMetodo1.addActionListener(new java.awt.event.ActionListener()
          {
        public void actionPerformed(java.awt.event.ActionEvent e) {
          if (jComboBoxMetodo1.getSelectedIndex() == 0){
            jLabel_aValue1.setEnabled(false);
            jTextField_aValue1.setEnabled(false);
          }
          else{
            jLabel_aValue1.setEnabled(true);
            jTextField_aValue1.setEnabled(true);
          }
        }
      });
    }
    return jComboBoxMetodo1;
  }
```

```
/**
 * This method initializes jButtonAGA1
 *
 * @return javax.swing.JButton
 */
private JButton getJButtonAGA1() {
  if (jButtonAGA1 == null) {
    jButtonAGA1 = new JButton();
    jButtonAGA1.setBounds(new Rectangle(340, 111, 92, 22));
    jButtonAGA1.setText("run AGA");
    jButtonAGA1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent e) {

        try{
          jc = new JFileChooser(pathOutputXML);
          ExampleFileFilter filtro = new ExampleFileFilter();
          filtro.addExtension("wgt");
          filtro.setDescription("ficheiro de output");
          jc.setFileFilter(filtro);
          jc.setDialogTitle("Gravar ficheiro output");
          jc.setSelectedFile(new File(path1.substring(path1.lastIndexOf("
              \\")+1, path1.lastIndexOf(".txt"))+".wgt"));
          jc.showSaveDialog(null);
          File file = jc.getSelectedFile();
          if (file!=null){
            pathOutputXML = file.getPath();
          }
          double pvalue = Double.parseDouble(jTextField_pvalue.getText());
          AlgoritmoAGA alg = new AlgoritmoAGA(path1, path2, pathOutputXML,
              pvalue);
          if (jComboBoxMetodo1.getSelectedIndex()==0)
            alg.listaISA();
          else
            alg.algoritmoAGA(Double.parseDouble(jTextField_aValue1.getText
                ()));
        }catch (IOException a){};
      }
    });
  }
  return jButtonAGA1;
}

/**
 * This method initializes jTextField_aValue1
 *
 * @return javax.swing.JTextField
 */
private JTextField getJTextField_aValue1() {
  if (jTextField_aValue1 == null) {
    jTextField_aValue1 = new JTextField();
    jTextField_aValue1.setBounds(new Rectangle(266, 114, 37, 20));
```

```
      jTextField_aValue1.setEnabled(false);
    }
    return jTextField_aValue1;
  }

  /**
   * This method initializes jTextField_pvalue
   *
   * @return javax.swing.JTextField
   */
  private JTextField getJTextField_pvalue() {
    if (jTextField_pvalue == null) {
      jTextField_pvalue = new JTextField();
      jTextField_pvalue.setSize(new Dimension(33, 23));
      jTextField_pvalue.setLocation(new Point(399, 83));
    }
    return jTextField_pvalue;
  }

  /**
   * @param args
   */
  public static void main(String[] args) {
    // TODO Auto-generated method stub
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        Interface application = new Interface();
        application.getJFrame().setVisible(true);
      }
    });
  }

  /**
   * This method initializes jFrame
   *
   * @return javax.swing.JFrame
   */
  private JFrame getJFrame() {
    if (jFrame == null) {
      jFrame = new JFrame();
      jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      jFrame.setSize(468, 203);
      jFrame.setResizable(false);
      jFrame.setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().
          getResource("/modelo2.jpg")));
      jFrame.setContentPane(getJTabbedPane());
      jFrame.setTitle("AGA");
      jFrame.setLocationRelativeTo(null);
    }
    return jFrame;
  }
```

```
  /**
   * This method initializes jContentPane1
   *
   * @return javax.swing.JPanel
   */
  private JPanel getJContentPane1() {
    if (jContentPane1 == null) {
      jLabelFicheiroXML = new JLabel();
      jLabelFicheiroXML.setText("Ficheiro XML");
      jLabelFicheiroXML.setSize(new Dimension(107, 21));
      jLabelFicheiroXML.setLocation(new Point(20, 86));
      jLabelFicheiro_a_Transformar = new JLabel();
      jLabelFicheiro_a_Transformar.setPreferredSize(new Dimension(67, 16));
      jLabelFicheiro_a_Transformar.setLocation(new Point(20, 55));
      jLabelFicheiro_a_Transformar.setSize(new Dimension(106, 21));
      jLabelFicheiro_a_Transformar.setText("Ficheiro original");
      jLabelGravarMatriz = new JLabel();
      jLabelGravarMatriz.setBounds(new Rectangle(18, 22, 156, 21));
      jLabelGravarMatriz.setText("Criar matriz");
      jContentPane1 = new JPanel();
      jContentPane1.setLayout(null);
      jContentPane1.add(jLabelGravarMatriz, null);
      jContentPane1.add(jLabelFicheiro_a_Transformar, null);
      jContentPane1.add(getJButtonFichTransformar(), null);
      jContentPane1.add(jLabelFicheiroXML, null);
      jContentPane1.add(getJButtonFichXML(), null);
      jContentPane1.add(getJButtonTransformar(), null);
    }
    return jContentPane1;
  }

}
```

Anexo 2 - Código XML do projecto RapidMiner utilizado

```
<operator name="Root" class="Process" expanded="yes">
    <parameter key="logfile" value="%{process_name}.log" />
    <operator name="OperatorChain" class="OperatorChain" expanded="yes">
        <operator name="ExampleSource" class="ExampleSource">
            <parameter key="attributes" value="%{process_name}.aml" />
        </operator>
        <operator name="AttributeWeightsLoader" class="
            AttributeWeightsLoader">
            <parameter key="attribute_weights_file" value="%{process_name}.
                wgt" />
        </operator>
        <operator name="AttributeWeightSelection" class="
            AttributeWeightSelection">
            <parameter key="weight" value="0.5" />
```

```
    </operator>
    <operator name="ExampleSetWriter" class="ExampleSetWriter">
        <parameter key="example_set_file" value="%{process_name}
            _AposFase1.dat"/>
        <parameter key="attribute_description_file" value="%{
            process_name}_AposFase1.aml"/>
    </operator>
</operator>
<operator name="OperatorChain (2)" class="OperatorChain" expanded="yes">
    <operator name="SVMWeighting" class="SVMWeighting">
    </operator>
    <operator name="AttributeWeightSelection (2)" class="
        AttributeWeightSelection">
        <parameter key="keep_attribute_weights" value="true"/>
        <parameter key="weight" value="0.55"/>
    </operator>
    <operator name="AttributeWeightsApplier" class="
        AttributeWeightsApplier">
    </operator>
    <operator name="ExampleSetWriter (2)" class="ExampleSetWriter">
        <parameter key="example_set_file" value="%{process_name}
            _AposFase2.dat"/>
        <parameter key="attribute_description_file" value="%{
            process_name}_AposFase2.aml"/>
    </operator>
</operator>
<operator name="FeatureSelection" class="FeatureSelection" expanded="yes
    ">
    <operator name="XValidation" class="XValidation" expanded="yes">
        <parameter key="keep_example_set" value="true"/>
        <parameter key="leave_one_out"  value="true"/>
        <operator name="LibSVMLearner" class="LibSVMLearner">
            <parameter key="kernel_type"  value="linear"/>
            <list key="class_weights">
            </list>
        </operator>
        <operator name="EvaluationChain" class="OperatorChain" expanded=
            "yes">
            <operator name="ModelApplier" class="ModelApplier">
                <list key="application_parameters">
                </list>
            </operator>
            <operator name="ClassificationPerformance" class="
                ClassificationPerformance">
                <parameter key="keep_example_set" value="true"/>
                <parameter key="classification_error" value="true"/>
                <list key="class_weights">
                </list>
            </operator>
        </operator>
    </operator>
</operator>
```

</operator>

Anexo 3 - Tabelas de resultados detalhados

Anaconda - ISA Greater

Método de correcção de *background*: Edwards

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 6 genes (64525, 60174, 59999, 60353, 140127, 61619)
	Com *a-value*=0.5	0.93% +/- 9.58% - 6 genes (64525, 60174, 59999, 60353, 140127, 61619)
	Com *a-value*=0.8	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
	Com *a-value*=0.95	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
	Com *a-value*=1	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
	Com *a-value*=0.5	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
	Com *a-value*=0.8	0.00% +/- 0.00% - 5 genes (63488, 60174, 20028, 62547, 61619)
	Com *a-value*=0.95	9.26% +/- 28.99% - 3 genes (16156, 60174, 140122)
	Com *a-value*=1	9.26% +/- 28.99% - 3 genes (16156, 60174, 140122)

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 6 genes (64525, 60174, 59999, 60353, 140127, 61619)
	Com *a-value*=0.5	0.93% +/- 9.58% - 6 genes (64525, 60174, 59999, 60353, 140127, 61619)
	Com *a-value*=0.8	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
	Com *a-value*=0.95	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
	Com *a-value*=1	0.93% +/- 9.58% - 3 genes (60174, 62787, 62019)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
	Com *a-value*=0.5	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
	Com *a-value*=0.8	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
	Com *a-value*=0.95	10.19% +/- 30.25% - 2 genes (63091, 15096)
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.5	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.8	1.85% +/- 13.48% 5 genes (63488, 63012, 62787, 16573, 67478)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (63012, 64753, 62787, 63467, 67478)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (63012, 64753, 62787, 63467, 67478)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	10.19% +/- 30.25% - 5 genes (66915, 63157, 60463, 62294, 140123)
	Com *a-value*=0.5	10.19% +/- 30.25% - 5 genes (66915, 63157, 60463, 62294, 140123)
	Com *a-value*=0.8	12.96% +/- 33.59% - 4 genes (63157, 65209, 59882, 140123)
	Com *a-value*=0.95	11.11% +/- 31.43% - 5 genes (63157, 65209, 59882, 61501, 140123)
	Com *a-value*=1	11.11% +/- 31.43% - 5 genes (63157, 65209, 59882, 61501, 140123)

Método de correcção de *background*: Half

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	9.26% +/- 28.99% - 3 genes (64615, 60174, 140122)
	Com *a-value*=0.5	9.26% +/- 28.99% - 3 genes (64615, 60174, 140122)
	Com *a-value*=0.8	9.26% +/- 28.99% - 3 genes (64615, 60174, 140122)
	Com *a-value*=0.95	9.26% +/- 28.99% - 3 genes (64615, 60174, 140122)
	Com *a-value*=1	9.26% +/- 28.99% - 3 genes (64615, 60174, 140122)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	2.78% +/- 16.43% - 4 genes (60174, 62019, 66986, 140127)
	Com *a-value*=0.5	0.93% +/- 9.58% - 6 genes (61557, 60174, 62512, 62019, 66986, 62129)
	Com *a-value*=0.8	8.33% +/- 27.64% - 4 genes (60174, 59902, 140122, 62125)
	Com *a-value*=0.95	8.33% +/- 27.64% - 4 genes (60174, 59902, 140122, 62125)
	Com *a-value*=1	4.63% +/- 21.01% - 4 genes (61273, 60174, 60094, 62342)

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	2.78% +/- 16.43% - 4 genes (63091, 63157, 62787, 62019)
	Com *a-value*=0.5	2.78% +/- 16.43% - 4 genes (63091, 63157, 62787, 62019)
	Com *a-value*=0.8	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
	Com *a-value*=0.95	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
	Com *a-value*=1	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	2.78% +/- 16.43% - 4 genes (64615, 64525, 63012, 61557)
	Com *a-value*=0.5	0.00% +/- 0.00% - 5 genes (64615, 63488, 60142, 63012, 62019)
	Com *a-value*=0.8	3.70% +/- 18.89% - 4 genes (63488, 63012, 62240, 67185)
	Com *a-value*=0.95	9.26% +/- 28.99% - 2 genes (63012, 67185)
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=0.8	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.95	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=1	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	14.81% +/- 35.52% - 3 genes (60606, 16261, 63446)
	Com *a-value*=0.5	14.81% +/- 35.52% - 3 genes (60606, 16261, 63446)
	Com *a-value*=0.8	14.81% +/- 35.52% - 3 genes (60606, 16261, 63446)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

Método de correcção de *background*: Minimum

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	3.70% +/- 18.89% - 4 genes (63082, 60174, 15638, 63452)
	Com *a-value*=0.5	3.70% +/- 18.89% - 4 genes (63082, 60174, 15638, 63452)
	Com *a-value*=0.8	2.78% +/- 16.43% - 5 genes (63082, 60174, 62990, 15638, 67478)
	Com *a-value*=0.95	2.78% +/- 16.43% - 5 genes (63082, 60174, 62990, 15638, 67478)
	Com *a-value*=1	2.78% +/- 16.43% - 5 genes (63082, 60174, 62990, 15638, 67478)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	1.85% +/- 13.48% - 4 genes (63268, 60174, 60010, 62547)
	Com *a-value*=0.5	1.85% +/- 13.48% - 4 genes (63268, 60174, 60010, 62547)
	Com *a-value*=0.8	2.78% +/- 16.43% - 4 genes (59879, 60174, 62787, 62547)
	Com *a-value*=0.95	2.78% +/- 16.43% - 4 genes (59879, 60174, 62787, 62547)
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 5 genes (60142, 60174, 62512, 62019, 63452)
	Com *a-value*=0.5	0.93% +/- 9.58% - 5 genes (60142, 60174, 62512, 62019, 63452)
	Com *a-value*=0.8	1.85% +/- 13.48% - 6 genes (60174, 59999, 61680, 15638, 63005, 61619)
	Com *a-value*=0.95	1.85% +/- 13.48% - 6 genes (60174, 59999, 61680, 15638, 63005, 61619)
	Com *a-value*=1	1.85% +/- 13.48% - 6 genes (60174, 59999, 61680, 15638, 63005, 61619)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.93% +/- 9.58% - 4 genes (60142, 63275, 60174, 62019)
	Com *a-value*=0.5	2.78% +/- 16.43% - 3 genes (61557, 60174, 60010)
	Com *a-value*=0.8	3.70% +/- 18.89% - 3 genes (62240, 62787, 14313)
	Com *a-value*=0.95	2.78% +/- 16.43% - 5 genes (15994, 15096, 62787, 59902, 63179)
	Com *a-value*=1	2.78% +/- 16.43% - 5 genes (15994, 15096, 62787, 59902, 63179)

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.8	0.00% +/- 0.00% - 7 genes (63012, 16579, 62787, 140223, 62019, 64419, 19273)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	19.44% +/- 39.58% - 3 genes (59870, 63446, 140123)
	Com *a-value*=0.5	19.44% +/- 39.58% - 3 genes (59870, 63446, 140123)
	Com *a-value*=0.8	7.41% +/- 26.19% - 2 genes (63012, 62787)
	Com *a-value*=0.95	37.04% +/- 48.29% - 1 gene (60463)
	Com *a-value*=1	37.04% +/- 48.29% - 1 gene (60463)

Método de correcção de *background*: Normexp

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 6 genes (64155, 59870, 60174, 67518, 140226, 15638)
	Com *a-value*=0.5	0.93% +/- 9.58% - 6 genes (64155, 59870, 60174, 67518, 140226, 15638)
	Com *a-value*=0.8	0.93% +/- 9.58% - 5 genes (64155, 60174, 67518, 140226, 15638)
	Com *a-value*=0.95	0.93% +/- 9.58% - 5 genes (64155, 60174, 67518, 140226, 15638)
	Com *a-value*=1	0.93% +/- 9.58% - 5 genes (64155, 60174, 67518, 140226, 15638)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	1.85% +/- 13.48% - 6 genes (62240, 60174, 63368, 15638, 63428, 140122)
	Com *a-value*=0.5	1.85% +/- 13.48% - 3 genes (62240, 60174, 60030)
	Com *a-value*=0.8	2.78% +/- 16.43% - 3 genes (62058, 60174, 62787)
	Com *a-value*=0.95	0.00% +/- 0.00% - 7 genes (15272, 60174, 62787, 60270, 62284, 19605, 63146)
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (63091, 59870, 60083, 62017, 62129)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (63091, 59870, 60083, 62017, 62129)
	Com *a-value*=0.8	2.78% +/- 16.43% - 4 genes (63091, 59870, 62017, 61680)
	Com *a-value*=0.95	2.78% +/- 16.43% - 4 genes (63091, 59870, 62017, 61680)
	Com *a-value*=1	2.78% +/- 16.43% - 4 genes (63091, 59870, 62017, 61680)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.93% +/- 9.58% - 6 genes (63091, 59870, 60083, 62017, 63048, 62129)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes(63091, 62130, 62787, 67383, 140123)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (63091, 62135, 62787, 67383, 140123)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (63091, 62135, 62787, 67383, 140123)
	Com *a-value*=1	12.04% +/- 32.54% - 2 genes (62357, 14847)

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=0.5	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=0.8	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=0.95	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=1	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	37.04% +/- 48.29% - 1 gene (67024)
	Com *a-value*=0.5	37.04% +/- 48.29% - 1 gene (67024)
	Com *a-value*=0.8	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

Anaconda - ISA Less

Método de correcção de *background*: Edwards

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	6.48% +/- 24.62% - 3 genes (60174, 62512, 15638)
	Com *a-value*=0.5	6.48% +/- 24.62% - 3 genes (60174, 62512, 15638)
	Com *a-value*=0.8	6.48% +/- 24.62% - 3 genes (60174, 62512, 15638)
	Com *a-value*=0.95	1.85% +/- 13.48% - 6 genes (66835, 63012, 62070, 61680, 15638, 62547)
	Com *a-value*=1	1.85% +/- 13.48% - 6 genes (66835, 63012, 62070, 61680, 15638, 62547)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.00% +/- 0.00% - 5 genes (63275, 60174, 62861, 16055, 62019)
	Com *a-value*=0.5	0.00% +/- 0.00% - 5 genes (63275, 60174, 62861, 16055, 62019)
	Com *a-value*=0.8	3.70% +/- 18.89% - 8 genes (60065, 60337, 63479, 22937, 63727, 65151, 16055, 62019)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	4.63% +/- 21.01% - 3 genes (15096, 15638, 60202)
	Com *a-value*=0.5	4.63% +/- 21.01% - 3 genes (15096, 15638, 60202)
	Com *a-value*=0.8	4.63% +/- 21.01% - 3 genes (15096, 15638, 60202)
	Com *a-value*=0.95	2.78% +/- 16.43% - 5 genes (16261, 15096, 59999, 60094, 15638)
	Com *a-value*=1	2.78% +/- 16.43% - 5 genes (16261, 15096, 59999, 60094, 15638)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
	Com *a-value*=0.5	1.85% +/- 13.48% - 3 genes (63091, 15096, 63275)
	Com *a-value*=0.8	9.26% +/- 28.99% - 3 genes (65151, 62019, 61508)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.5	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	11.11% +/- 31.43% - 6 genes (60142, 62350, 66959, 59948, 67078, 62294)
	Com *a-value*=0.5	11.11% +/- 31.43% - 6 genes (60142, 62350, 66959, 59948, 67078, 62294)
	Com *a-value*=0.8	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

Método de correcção de *background*: Half

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 6 genes (60174, 62787, 15638, 62894, 62547, 63259)
	Com *a-value*=0.5	0.93% +/- 9.58% - 6 genes (60174, 62787, 15638, 62894, 62547, 63259)
	Com *a-value*=0.8	0.93% +/- 9.58% - 6 genes (60174, 62787, 15638, 62894, 62547, 63259)
	Com *a-value*=0.95	0.93% +/- 9.58% - 5 genes (63023, 63012, 62787, 62070, 63259)
	Com *a-value*=1	0.93% +/- 9.58% - 5 genes (63023, 63012, 62787, 62070, 63259)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.93% +/- 9.58% - 5 genes (62921, 60174, 60030, 62205, 140123)
	Com *a-value*=0.5	0.93% +/- 9.58% - 5 genes (62921, 60174, 60030, 62205, 140123)
	Com *a-value*=0.8	6.48% +/- 24.62% - 6 genes (22937, 64753, 60353, 64296, 67191, 62294)
	Com *a-value*=0.95	15.74% +/- 36.42% - 4 genes (60020, 62368, 64665, 67193)
	Com *a-value*=1	15.74% +/- 36.42% - 4 genes (60020, 62368, 64665, 67193)

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 5 genes (63023, 63012, 61018, 62787, 62070)
	Com *a-value*=0.5	0.93% +/- 9.58% - 5 genes (63023, 63012, 61018, 62787, 62070)
	Com *a-value*=0.8	0.93% +/- 9.58% - 5 genes (63023, 63012, 61018, 62787, 62070)
	Com *a-value*=0.95	9.26% +/- 28.99% - 2 genes (63012, 59870)
	Com *a-value*=1	9.26% +/- 28.99% - 2 genes (63012, 59870)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.93% +/- 9.58% - 4 genes (63012, 59870, 63275, 140122)
	Com *a-value*=0.5	0.93% +/- 9.58% - 4 genes (63012, 59870, 63275, 140122)
	Com *a-value*=0.8	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (63012, 62839, 62787, 62019, 64419)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (63012, 62839, 62787, 62019, 64419)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (63012, 62839, 62787, 62019, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	2.78% +/- 16.43% - 8 genes (64235, 63284, 64753, 59890, 20028, 60178, 59948, 61619)
	Com *a-value*=0.5	2.78% +/- 16.43% - 8 genes (64235, 63284, 64753, 59890, 20028, 60178, 59948, 61619)
	Com *a-value*=0.8	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

Método de correcção de *background*: Minimum

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 7 genes (66835, 63268, 63012, 19587, 60022, 61619, 140123)
	Com *a-value*=0.5	0.93% +/- 9.58% - 7 genes (66835, 63268, 63012, 19587, 60022, 61619, 140123)
	Com *a-value*=0.8	0.93% +/- 9.58% - 7 genes (66835, 63268, 63012, 19587, 60022, 61619, 140123)
	Com *a-value*=0.95	0.93% +/- 9.58% - 7 genes (66835, 63268, 63012, 19587, 60022, 61619, 140123)
	Com *a-value*=1	0.93% +/- 9.58% - 7 genes (66835, 63268, 63012, 19587, 60022, 61619, 140123)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.93% +/- 9.58% - 4 genes (62240, 63275, 60174, 63428)
	Com *a-value*=0.5	0.93% +/- 9.58% - 4 genes (62240, 63275, 60174, 63428)
	Com *a-value*=0.8	7.41% +/- 26.19% - 6 genes (64615, 60353, 65151, 61127, 62019, 64419)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	9.26% +/- 28.99% - 2 genes (64618, 15994)
	Com *a-value*=0.5	9.26% +/- 28.99% - 2 genes (64618, 15994)
	Com *a-value*=0.8	9.26% +/- 28.99% - 2 genes (64618, 15994)
	Com *a-value*=0.95	9.26% +/- 28.99% - 2 genes (64618, 15994)
	Com *a-value*=1	9.26% +/- 28.99% - 2 genes (64618, 15994)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	0.93% +/- 9.58% - 4 genes (63275, 60174, 62019, 61508)
	Com *a-value*=0.5	0.93% +/- 9.58% - 4 genes (63275, 60174, 62019, 61508)
	Com *a-value*=0.8	8.33% +/- 27.64% - 5 genes (59987, 65151, 62019, 64296, 61508)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	11.11% +/- 31.43% - 8 genes (60142, 61101, 61510, 64580, 63120, 59948, 67189, 62392)
	Com *a-value*=0.5	11.11% +/- 31.43% - 8 genes (60142, 61101, 61510, 64580, 63120, 59948, 67189, 62392)
	Com *a-value*=0.8	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

Método de correcção de *background*: Normexp

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (59870, 60174, 13824, 15638, 63048)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (59870, 60174, 13824, 15638, 63048)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (59870, 63177, 61680, 63048, 14313)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (59870, 63177, 61680, 63048, 14313)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (59870, 63177, 61680, 63048, 14313)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	1.85% +/- 13.48% - 4 genes (16055, 60030, 14313, 62129)
	Com *a-value*=0.5	1.85% +/- 13.48% - 4 genes (16055, 60030, 14313, 62129)
	Com *a-value*=0.8	10.19% +/- 30.25% - 4 genes (64615, 67210, 60353, 67191)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (63091, 62130, 59870, 62240, 63048)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (63091, 62130, 59870, 62240, 63048)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (63082, 63091, 59870, 62240, 63048)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (63082, 63091, 59870, 62240, 63048)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (63082, 63091, 59870, 62240, 63048)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	2.78% +/- 16.43% - 3 genes (63488, 63091, 63275)
	Com *a-value*=0.5	2.78% +/- 16.43% - 3 genes (63488, 63091, 63275)
	Com *a-value*=0.8	4.63% +/- 21.01% - 6 genes (59871, 22937, 67518, 140223, 65151, 64296)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	5.56% +/- 22.91% - 3 genes (59871, 63091, 62787)
	Com *a-value*=0.5	5.56% +/- 22.91% - 3 genes (59871, 63091, 62787)
	Com *a-value*=0.8	3.70% +/- 18.89% - 5 genes (63091, 63012, 62839, 62787, 64514)
	Com *a-value*=0.95	3.70% +/- 18.89% - 5 genes (63091, 63012, 62839, 62787, 64514)
	Com *a-value*=1	3.70% +/- 18.89% - 5 genes (63091, 63012, 62839, 62787, 64514)
ISA-$Q_{\frac{1}{2}}$	Sem Selecção	18.52% +/- 38.84% - 7 genes (60716, 62076, 63354, 63112, 16274, 140191, 62311)
	Com *a-value*=0.5	21.30% +/- 40.94% - 5 genes (60256, 63112, 63132, 16274, 140188)
	Com *a-value*=0.8	37.04% +/- 48.29% - 1 gene (65204)
	Com *a-value*=0.95	sem resultados após aplicação dos pesos no RapidMiner
	Com *a-value*=1	sem resultados após aplicação dos pesos no RapidMiner

Anaconda - ISA Module

Método de correcção de *background*: Edwards

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	12.04% +/- 32.54% - 4 genes (60232, 69747, 61594, 62277)
	Com *a-value*=0.5	12.04% +/- 32.54% - 4 genes (60232, 69747, 61594, 62277)
	Com *a-value*=0.8	5.56% +/- 22.91% - 5 genes (63573, 16261, 15638, 60763, 60121)
	Com *a-value*=0.95	5.56% +/- 22.91% - 5 genes (63573, 16261, 15638, 60763, 60121)
	Com *a-value*=1	5.56% +/- 22.91% - 5 genes (63573, 16261, 15638, 60763, 60121)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	5.56% +/- 22.91% - 5 genes (16261, 62966, 60763, 60883, 63262)
	Com *a-value*=0.5	5.56% +/- 22.91% - 5 genes (16261, 62966, 60763, 60883, 63262)
	Com *a-value*=0.8	5.56% +/- 22.91% - 5 genes (16261, 62966, 60763, 60883, 63262)
	Com *a-value*=0.95	5.56% +/- 22.91% - 5 genes (16261, 62966, 60763, 60883, 63262)
	Com *a-value*=1	5.56% +/- 22.91% - 5 genes (16261, 62966, 60763, 60883, 63262)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.5	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.8	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=0.95	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
	Com *a-value*=1	0.93% +/- 9.58% - 5 genes (63012, 62787, 140122, 67478, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

Método de correcção de *background*: Half

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	3.70% +/- 18.89% - 4 genes (66835, 15994, 60763, 62294)
	Com *a-value*=0.5	3.70% +/- 18.89% - 4 genes (66835, 15994, 60763, 62294)
	Com *a-value*=0.8	12.04% +/- 32.54% - 2 genes (60064, 15994)
	Com *a-value*=0.95	12.04% +/- 32.54% - 2 genes (60064, 15994)
	Com *a-value*=1	12.04% +/- 32.54% - 2 genes (60064, 15994)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	3.70% +/- 18.89% - 5 genes (15994, 59893, 62468, 15638, 61619)
	Com *a-value*=0.5	3.70% +/- 18.89% - 5 genes (15994, 59893, 62468, 15638, 61619)
	Com *a-value*=0.8	3.70% +/- 18.89% - 5 genes (15994, 59893, 62468, 15638, 61619)
	Com *a-value*=0.95	3.70% +/- 18.89% - 5 genes (15994, 59893, 62468, 15638, 61619)
	Com *a-value*=1	3.70% +/- 18.89% - 5 genes (15994, 59893, 62468, 15638, 61619)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

Método de correcção de *background*: Minimum

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	2.78% +/- 16.43% - 7 genes (15994, 67129, 61799, 60032, 63375, 63048, 67478)
	Com *a-value*=0.5	2.78% +/- 16.43% - 7 genes (15994, 67129, 61799, 60032, 63375, 63048, 67478)
	Com *a-value*=0.8	9.26% +/- 28.99% - 3 genes (15592, 15994, 63832)
	Com *a-value*=0.95	9.26% +/- 28.99% - 3 genes (15592, 15994, 63832)
	Com *a-value*=1	9.26% +/- 28.99% - 3 genes (15592, 15994, 63832)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	9.26% +/- 28.99% - 2 genes (64618, 15994)
	Com *a-value*=0.5	9.26% +/- 28.99% - 2 genes (64618, 15994)
	Com *a-value*=0.8	3.70% +/- 18.89% - 7 genes (66835, 60715, 15994, 16579, 62512, 140123, 62125)
	Com *a-value*=0.95	3.70% +/- 18.89% - 7 genes (66835, 60715, 15994, 16579, 62512, 140123, 62125)
	Com *a-value*=1	3.70% +/- 18.89% - 7 genes (66835, 60715, 15994, 16579, 62512, 140123, 62125)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (59871, 63012, 62787, 62019, 64419)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (60064, 63012, 62787, 62019, 64419)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

Método de correcção de *background*: Normexp

	Método de Normalização	*IgloessSlloess*
ISA-$\bar{X}$	Sem Selecção	0.00% +/- 0.00% - 7 genes (66835, 15820, 15994, 13824, 60083, 64071, 63247)
	Com *a-value*=0.5	0.00% +/- 0.00% - 7 genes (66835, 15820, 15994, 13824, 60083, 64071, 63247)
	Com *a-value*=0.8	1.85% +/- 13.48% - 5 genes (66835, 15994, 61656, 60083, 67191)
	Com *a-value*=0.95	1.85% +/- 13.48% - 5 genes (66835, 15994, 61656, 60083, 67191)
	Com *a-value*=1	1.85% +/- 13.48% - 5 genes (66835, 15994, 61656, 60083, 67191)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	*IlloessSlloess*
ISA-$\bar{X}$	Sem Selecção	1.85% +/- 13.48% - 5 genes (64155, 63023, 63091, 62063, 64580)
	Com *a-value*=0.5	1.85% +/- 13.48% - 5 genes (64155, 63023, 63091, 62063, 64580)
	Com *a-value*=0.8	4.63% +/- 21.01% - 4 genes (63091, 60174, 62017, 67191)
	Com *a-value*=0.95	1.85% +/- 13.48% - 7 genes (59871, 63091, 66835, 62063, 67518, 62125, 62129)
	Com *a-value*=1	1.85% +/- 13.48% - 7 genes (59871, 63091, 66835, 62063, 67518, 62125, 62129)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	

	Método de Normalização	NoNorm
ISA-$\bar{X}$	Sem Selecção	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=0.5	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=0.8	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=0.95	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
	Com *a-value*=1	4.63% +/- 21.01% - 4 genes (63091, 62240, 62787, 62019)
ISA-$Q_{\frac{1}{2}}$	Sem resultados	